Titre: Title:	Protocoles de coopératon dans les systèmes multiagents : une approche basée sur les relations de dépendances
Auteur: Author:	Sidi Larbi Esmahi
Date:	2000
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Esmahi, S. L. (2000). Protocoles de coopératon dans les systèmes multiagents : une approche basée sur les relations de dépendances [Ph.D. thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8541/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8541/
Directeurs de recherche: Advisors:	
Programme: Program:	Unspecified

UNIVERSITÉ DE MONTRÉAL

PROTOCOLES DE COOPÉRATION DANS LES SYSTEMES MULTIAGENTS : UNE APPROCHE BASÉE SUR LES RELATIONS DE DÉPENDANCES

ESMAHI SIDI LARBI

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIAE DOCTOR (Ph. D.)

(GÉNIE ÉLECTRIQUE)

Mai 2000

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

PROTOCOLES DE COOPÉRATION DANS LES SYSTEMES MULTIAGENTS : UNE APPROCHE BASÉE SUR LES RELATIONS DE DÉPENDANCES

Présentée par : Esmahi Sidi Larbi

En vue de l'obtention du diplôme de : Philosophiae Doctor

A été dûment acceptée par le jury d'examen constitué de :

M. Desantis Romano, PhD., président

M. Bernard Jean-Charles, PhD., membre et directeur de recherche

M. Dini Petre, PhD., membre et codirecteur de recherche

M. Pierre Samuel, PhD., membre

M. Esfandiari Babak, PhD., membre externe

Dédicace

À la mémoire de ma mère,

à mon père,

au petit qui vit encore pour quelques mois dans les trois obscurités,

à ma femme, frères et sœurs.

Remerciements

Je tiens tout d'abord à remercier mon directeur de recherche, Dr Jean-Charles Bernard et mes codirecteurs, Dr Petre Dini et M Daniel Gauvin, pour leurs conseils toujours pertinents, leur support, leur disponibilité et leur chaleur humaine. Je n'aurais pu espérer un meilleur encadrement de leur part, car ils m'ont toujours laissé toute la latitude désirée dans mon projet tout en me fournissant les directives nécessaires à un bon avancement.
Je veux également souligner ma reconnaissance envers les membres du jury, Messieurs, Dr Romano Desantis, Dr Samuel Pierre, et Dr Babak Esfandiari pour le temps qu'ils ont consacré à la lecture et à la critique de cette thèse, ainsi qu'envers le représentant du doyen de l'Université de Montréal Monsieur Dr Patrick d'Astous.

Je suis également très reconnaissant envers mes parents et tous les membres de ma famille qui, tout au long de mon projet d'études, étaient présents pour me supporter et m'encourager à pousser mes capacités à leurs limites.
Je ne saurais oublier les amis que j'ai côtoyés au cours de mon stage au Centre de Recherche Informatique de Montréal (CRIM) au consortium MIAMI et au département de génie électrique et génie informatique, Dr Stefan Covacci, Dr Gilles Roy, Hervé Marchal, Jean-François Rizand et Ming Bai qui m'ont permis de travailler dans une ambiance de recherche formidable.

Je tiens également à faire part de mon amitié et ma considération envers toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce travail. Un salut particulier à mes amis au regroupement des marocains au Canada (RMC) qui, par leur présence, leurs encouragements, leur bonne humeur, leurs conseils et leur amour ont rendu ces années très agréables et enrichissantes.

Je suis très redevable envers le Centre de Recherche Informatique de Montréal (CRIM), le Conseil de Recherches en Sciences et en Génie du Canada (CRSNG) et la fondation SONY pour le support financier qu'ils m'ont accordé.

Finalement, je désire adresser un merci bien spécial à mon épouse Ikram Cheikhi qui a accepté de vivre les conditions difficiles de ces années. Sans ses sacrifices, son amour et son appui inconditionnel, la production de cette thèse aurait été une tâche beaucoup plus ardue.

À toutes ces personnes et organisations et à celles et ceux qui m'auraient aidé de près ou de loin et que j'aurais oubliés mille merci !

Résumé

L'avancement conjugué des technologies de télécommunication et des structures des organisations a conduit à l'émergence de nouvelles structures d'information, en l'occurrence les entreprises virtuelles et les marchés virtuels. Ces nouvelles structures sont modélisées sous forme d'un ensemble d'agents logiciels intelligents qui interagissent, coopèrent ou concurrencent dans un univers multiagent ouvert.

Cette thèse présente un modèle de coopération pour systèmes multiagent fondé sur la théorie de l'utilité, les relations de dépendance et le raisonnement par cas. Ce modèle permet à un agent de raisonner sur ses pairs et plus particulièrement de déterminer ses relations et situations de dépendance et d'évaluer le profit relatif à une coalition. Nous considérons l'intégration d'un mécanisme de coopération, comme une composante essentielle dans la conception d'agents autonomes évoluant dans un système multiagent ouvert. La notion d'ouverture est fondamentale pour les applications considérées dans cette thèse : entreprises virtuelles et places commerciales virtuelles. L'ouverture du système multiagent dans ce contexte désigne la capacité d'ajouter ou de retirer dynamiquement des agents dans le système.

Comme dans ces systèmes l'organisation des agents ne peut pas être spécifiée pendant la phase de conception, la coopération est fondée sur la formation dynamique de coalition. Comme nous ne supposons pas que les agents soient bienveillants, notre modèle intègre une stratégie de négociation basée sur les fonctions d'utilité et l'évaluation du risque.

Pour valider le modèle proposé nous avons implanté des prototypes sur les plates-formes MIAMI et LALO. Principalement, la création d'entreprise virtuelle et la place commerciale virtuelle MIAMAP nous a permis de faire des évaluations expérimentales.

Les résultats de simulation montrent que le modèle permet de faire un équilibre entre le besoin de rationalité individuelle des agents et le besoin de cohérence sociale du système. En fait, la conséquence la plus évidente de l'utilisation de ce modèle est sa capacité d'améliorer l'engagement social des agents.

Cette thèse contribue à trois domaines inter-reliés de l'intelligence artificielle distribuée : l'interaction des agents intelligents, la négociation et résolution de conflits, et la formation de coalitions.

D'un point de vue modélisation de l'interaction dans les SMA, nous avons procédé à une analyse de l'interaction dans les SMA selon les principaux facteurs qui influencent le comportement des agents : la situation de l'agent par rapport à l'interaction, les relations bilatérales entre les agents et l'état mental de l'agent pendant l'interaction. Suite à cette analyse et à la critique de l'état de l'art, nous avons proposé un modèle utilisant les notions d'influence et d'adoption de buts, permettant ainsi une amélioration majeure aux approches usuelles fondées exclusivement sur les notions d'utilité.

Pour ce qui est de la négociation et la résolution de conflits nous avons introduit une méthode combinant l'utilisation de la stratégie de négociation à base d'évaluation de risque et l'utilisation des relations de dépendances entre les agents. Cela a entraîné deux conséquences importantes : d'une part, l'élargissement de l'ensemble des ententes possibles entre les agents et d'autre part l'émergence d'un équilibre entre la performance individuelle des agents et la performance globale du système.

En ce qui concerne la formation de coalitions, notre modèle introduit d'une part, la notion de persuasion pour permettre à un agent de convaincre, par argumentation, ses pairs à adopter ses buts et d'autre part, la notion d'engagement social permettant le maintient d'un équilibre entre la rationalité individuelle et la cohérence sociale dans le système.

Abstract

The combined advance of telecommunication technologies and organizations structures implies the emergence of new information structures. In fact, virtual enterprises and virtual marketplaces are significant examples. These new structures are modeled as using intelligent software agents who interact, cooperate and compete in an open multiagent world.

This thesis presents a co-operation model for multiagent systems based on the utility theory, the dependency relations and the case based reasoning. The model allows an agent to reason on its peers, in particular to determine its relations and situations of dependence and to evaluate the profit relating to a coalition. We consider the integration of a co-operation mechanism, as an essential component in the design of autonomous agents, which are evolving in an open multiagent system. The concept of opening is fundamental for the applications considered in this thesis: virtual enterprises and virtual market-places. By open we mean that agents may enter or leave the agency at any moment.

As in such systems, the organization of the agents can't be conceived during the design phase, the co-operation is based on the dynamic formation of coalition. As we don't suppose that agents are benevolent, our model integrates a negotiation strategy using utility functions and risk evaluation.

To validate the model we have implemented prototypes on the MIAMI and LALO platforms. Mainly, the virtual enterprise building and the virtual market-place MIAMAP have been used for experimental evaluations.

The experimental results show that the model makes a balance between the need of the agents' individual rationality and the need for social coherence in the overall system. In fact, the most obvious consequence for the use of this model, is its capacity to improve the agents' social commitment.

This thesis brings contribution to three inter-linked domains in distributed artificial intelligence: agents' interaction, automated negotiation and conflict resolution, and coalition formation.

From the interaction modeling point of view, the model which we proposed allows to take into account the concept of influence and goals adoption, things absent in the approaches based exclusively on utility.

In regards to the negotiation and resolution of conflicts, we have used a combination of a negotiation strategy based on risk evaluation and the agents' relations of dependency. This combination has led to two significant consequences: on one hand the widening of the set of possible agreements between the agents and on the other hand the emergence of a balance between the individual performance of the agents and the global performance of the system.

With regard to the coalition formation, our model introduces two concepts: the persuasion and the social commitment. The persuasion concept allows an agent to convince by argumentation its partners to adopt its goals and the concept of social commitment allows making a balance between individual rationality and social coherence in the system.

Table des matières

Liste des tableaux

Liste des figures

Liste des acronymes

ACL	Agent Communication Language
ACTS	Advanced Communications Technologies Studies
AVP	Active Virtual Pipe
CBR	Case Based Reasoning
CDN	Constraint-Directed Negotiation
CLIMATE	Cluster for Intelligent Mobile Agents for Telecommunication Environments
CNP	Contract-Net Protocol
COOL	Coordination Language
CORBA	Common Object Request Broker Architecture
CP	Connectivity Provider
DPS	Distributed Problem Solver
FIPA	Foundation for Intelligent Physical Agents
FIPA-ACL	FIPA's Agent Communication Language
IAD	Intelligence Artificielle Distribuée
KBN	Knowledge Based Negotiation
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation Language
LALO	Langage Agent Logiciel Objet
MAS	Multi-Agents System
MASIF	Mobile Agent System Interoperability Facility
MIA	Mobile Intelligent Agent
MIAMAP	MIAMI Market Place
MIAMI	Mobile Intelligent Agents for Managing Information Infrastructure
MSN	MultiStage Negotiation

OMG	Object Management Group
RDP	Résolution Distribuée de Problèmes
RP	Remote Programming
RPC	Remote Procedure Call
SAP	Sharable Agent Perspectives
SMA	Systèmes Multiagent
VAE	Vente aux Enchères
VE	Virtual Enterprise
VEB	Virtual Enterprise Builder

Chapitre 1: Introduction

1.1 Motivations pour ce travail

L'expansion des réseaux, l'Internet, ainsi que l'avancement des technologies de télécommunication font de plus en plus exploser la quantité de services et de tâches que nous pouvons obtenir ou réaliser à partir de nos ordinateurs. Par conséquent, la combinaison du commerce électronique et des agents intelligents (cf. 2.2) offre un potentiel d'énormes profits économiques[1] [IDC 1999a, RCC 1999]. Plusieurs applications sont déjà opérationnelles sur Internet, sous formes de places commerciales virtuelles [Benjamin 1995, Zwass 1999, VME], agents courtiers virtuels [Fikes 1995, Keller 1995] ou services de ventes aux enchères [ebay, OnSale]. Cependant, plusieurs de ces projets sont loin de supporter des activités économiques complexes comme la négociation de contrats et l'exécution automatique de transaction ou l'élaboration et la mise en opération d'entreprises virtuelles.

Le thème central de cette thèse porte sur l'intégration des facilités de négociation dans ces systèmes complexes que sont : les places commerciales et les entreprises virtuelles. La nécessité d'y intégrer des mécanismes de négociation automatique est devenue pressante en raison de quatre développements majeurs dans les technologies de l'information et les structures des organisations. Premièrement, l'évolution conjuguée de la technologie des télécommunications et de la structure des organisations conduit à l'émergence de nouvelles structures réparties et complexes que les auteurs appellent 'sociétés d'agents' [Minsky 1985], 'sociétés d'objets' [Hewitt 1993, Nicol 1993] ou 'entreprises virtuelles' [Dewey 1999]. Les environnements de traitement de l'information dans ces structures sont composés de grands réseaux de ressources de calcul hétérogènes,

[1] Les chiffres donnés en 1999 par IDC (International Data Corporation) dans les références citées ci-dessus, estiment que le commerce électronique se chiffre au Canada à 10.95 Milliard de $CAN et au USA à 74.43 Milliard de $US. Les prévisions pour l'an 2003 dépassent les 80.30 Milliard de $CAN au Canada et 707.91 Milliard de $US au USA.

autonomes, distribués et ouverts. Par conséquent, il faut d'abord assurer l'interconnexion et l'interopérabilité de leurs composantes, ainsi que la coopération et l'adaptation de ces derniers aux changements pouvant avoir lieu à cause de l'ajout ou la suppression dynamique de services. Deuxièmement, l'évolution très rapide des petites transactions de commerce via l'Internet[2] [IDC 1999b] pour l'achat de biens, services d'information ou services de communication [Sandholm 1995, Esmahi 1999b], conduit à la nécessité d'automatiser la réalisation de telles transactions. Troisièmement, l'avancement réalisé ces trois dernières années sur la standardisation des langages de communications (KQML[3] [Finin 1994, Labrou 1994, Labrou 1997], FIPA-ACL [FIPA 1997a, FIPA 1998], KIF [Genesereth 1992], COOL [Barbuceanu 1996a, Barbuceanu 1996b]) a ouvert la voie pour faire interagir, dans des environnements ouverts et en temps réel, des agents appartenant à différentes organisations et conçus selon différentes architectures. Quatrièmement, nous croyons, comme annoncé par Negroponte [Negroponte 1995] que dans le futur, le traitement informatique va se faire de plus en plus par délégation, et non pas par la possession ou la manipulation directe de machines super-puissantes. Ainsi, pour réaliser une telle fonctionnalité, les systèmes informatiques doivent être : autonomes, proactifs, coopératifs et adaptables (cf. 2.2). Étant donné qu'un service a un coût associé, nous devons concevoir les moyens à travers lesquels un système (agent) peut vouloir accepter de fournir un service ou de coopérer avec d'autres systèmes, puisque dans ce contexte-ci, la coopération ne peut pas être prise en compte comme une hypothèse de départ.

1.2 Contexte et problématique

Nos recherches concernent la construction et le fonctionnement de systèmes multiagent (SMA) [Bond 1988]. Elles se focalisent sur le problème de la coopération et de

[2] Selon les chiffres de IDC dans les références ci-dessus, le nombre de personnes qui font des achats par Internet va augmenter de 31 Millions en 1998 pour atteindre 183 Millions en 2003. Plus de la moitié de cette population est dans l'Amérique du nord.

[3] Tous les mots acronymes sont définis dans le glossaire de ce document.

l'interopérabilité des agents composant de tels systèmes. Le cadre d'application de nos travaux est la gestion des services de télécommunication.

Le domaine des télécommunications est un cadre d'application qui illustre pleinement les difficultés du problème de coopération. Un système de télécommunication est un système réparti complexe du fait de sa portée géographique étendue, de sa grande hétérogénéité tant matérielle que logicielle, de sa dimension temps-réel ou encore de sa contrainte forte de fonctionnement continu. Les applications qu'il supporte (services de télécommunication) doivent répondre à des exigences de qualité de service telles que la disponibilité, la fiabilité, la sécurité et le coût. La construction de SMA ouverts dans un tel contexte doit alors intégrer l'ensemble de ces caractéristiques et contraintes.

Le processus de construction de ces systèmes répartis est fondé sur une approche par composantes. En effet, un système réparti est constitué d'un assemblage de composantes qui, à travers leurs interactions, fournissent la fonctionnalité attendue. Sa construction vise alors à définir son architecture en décrivant d'une part ses composants et d'autre part leur assemblage selon un schéma d'interaction propre à l'application. Le problème est double. Il faut d'une part disposer de composantes ayant de réelles caractéristiques de flexibilité pour s'adapter à l'assemblage et d'autre part s'assurer qu'une fois l'assemblage formé, il fournit bien la fonctionnalité attendue.

Les agents logiciels sont des composants répondant à l'exigence de flexibilité énoncée ci-dessus. En effet, un agent est une entité logicielle qui exhibe des capacités d'interactions et d'adaptation. Le paradigme agent permet de construire un système sous la forme d'un ensemble d'agents coopérants, c'est à dire d'un système multiagent. Il offre un niveau de modélisation qui facilite l'intégration d'une application dans un système opérationnel, la combinaison d'applications et l'automatisation de l'usage des applications. Il reste cependant un grand défi, qui consiste à intégrer aux composantes du système un mécanisme de coopération qui, sans mettre en cause leur autonomie, permet au système de fonctionner de façon optimale.

Un autre concept du paradigme agent, plus attrayant que les précédents pour ce contexte, est celui de la mobilité qui permet aux agents de se déplacer dans un réseau sur des sites

distants pour y exécuter des tâches spécifiques [Chess 1995, Harrison 1995]. En effet, avec des agents mobiles, on peut simplifier davantage la modélisation et le développement du système en faisant la partition des activités et fonctionnalités sur des entités autonomes (agents), tout en maintenant à un niveau acceptable la charge de communication entre ces entités. La mobilité des agents est une étape de plus dans la simplification de la construction du système, puisqu'elle permet de faire un équilibre entre l'échange de messages et l'envoi du code.

La mobilité se fonde sur un principe de communication appelé «programmation distante» (*Remote Programming — RP*) [Magedanz 1996, White 1996], jugé plus économique que l'appel de procédure à distance (*Remote Procedure Call — RPC*) [Birrel 1984] car l'agent interagit localement plutôt qu'à distance avec les ressources nécessaires à l'accomplissement de sa tâche. Pour supporter la mise en œuvre de cette mobilité, des plates-formes agents ont vu le jour, la plupart d'entre elles se fondant sur Java [Fisher 1994, Kotz 1997, Straber 1997]. *Aglet* [Lange 1998a, Lange 1998b], *Concordia* [Wong 1999], *Grasshopper* [IKV 1997] ou *Voyager* [Objectspace 1997] sont des exemples commercialisés de telles plates-formes. Les exemples types d'applications orientées agent (mobile ou non) développées au-dessus de ces systèmes sont le commerce électronique [Merz 1996, Villinger 1997], la recherche d'informations [Bowman 1994, Debra 1994, Perret 1996], la gestion de réseau [Esfandiari 1996, Oliveira 1996, Omari 1999] ou le multimédia [Falchuck 1997, Nygren 1996, Smith 1999].

La technologie agent faisant l'objet de nombreux travaux, une intégration des concepts clés est apparue essentielle avec comme finalité la production de spécifications destinées à l'industrie. C'est l'objectif que s'est fixé la *Foundation for Intelligent Physical Agents* (FIPA) [FIPA 1997b], c'est à dire de promouvoir le développement et la spécification de la technologie agent. De même l'*Object Management Group* (OMG) [OMG 1994] a étendu le champ de ses travaux initialement fondés sur le paradigme objet pour y intégrer la technologie agent mobile en vue de produire des spécifications.

En télécommunication, les premiers travaux utilisant la technologie agent se sont focalisés sur la gestion de réseau par délégation [Yemeni 1991, Znaty 1997] et sur la

fourniture d'assistants personnels [Kozbe 1996, Reinhardt 1994]. Aujourd'hui, l'effort porte sur le développement de plates-formes permettant l'implantation de services de télécommunication fondés sur des agents [Tennenhouse 1997, Magedanz 1999]. Afin de coordonner ces recherches, le programme européen ACTS a constitué en février 1998 la banque de données CLIMATE (*Cluster for Intelligent Mobile Agents for Telecommunication Environments*) [CLIMATE 1997] et a lancé plusieurs projets [ACTS 1997].

Le projet MIAMI[4], avec une problématique axée sur le domaine des agents mobiles intelligents, est l'un des plus importants de ces projets. Cette thèse a été menée en grande partie dans le cadre applicatif de la plate-forme MIAMI. Les premiers travaux ont été réalisés dans le cadre d'un autre projet qui est LALO[5] [Gauvin 1995, Gauvin 1996, Esmahi 1996, Bernard 1998].

Le projet MIAMI a pour objectifs de : (1) développer une plate-forme unifiée pour agents mobiles, en se basant sur les spécifications d'interopérabilité MASIF[6] [MASIF 1998], (2) développer des solutions, par agents mobiles, pour la gestion des structures d'information[7] complexes et valider ces solutions par des scénarios d'application, (3) créer une référence d'implantation pour les plates-formes d'agents mobiles, (4) produire des recommandations pour l'amélioration des standards MASIF.

Les deux principales applications développées dans le cadre du projet MIAMI sont: (1) le service de canal de communication virtuel *(Active Virtual Pipe — AVP)* qui démontre l'intérêt d'utilisation de la technologie d'agents mobiles pour la gestion des services de communications, et (2) le service d'entreprise virtuelle *(Virtual Entreprise Builder — VEB)* qui démontre l'intérêt d'utilisation des agents intelligents pour le commerce électronique, la négociation de contrats et les services de traitement de l'information.

---

[4] MIAMI pour designer '*Mobile Intelligent Agents for Managing the Information Infrastructure*'. Plus de détails sont disponibles au : http://www.fokus.gmd.de/research/ cc/ima/miami/entry.html

[5] LALO : pour designer '*Langage Agent Logiciel Objet*'. LALO est un environnement pour la programmation orientée agents qui a été développé au Centre de Recherche Informatique de Montréal (CRIM). Pour plus d'information voir le site : http://www.crim.ca/sbc/francais/lalo/index.html.

[6] MASIF pour designer "*Mobile Agent System Interoperability Facility*". Pour plus d'information consulter l'adresse : http://www.fokus.gmd.de/cc/ima/masif/

[7] Le contexte du projet MIAMI est l'infrastructure d'information européenne, qui est de plus en plus ouverte et caractérisée par sa distribution, sa dynamique et la complexité de ses ressources.

La figure 1.1 montre les principales composantes de la plate-forme MIAMI [Guther 1999].

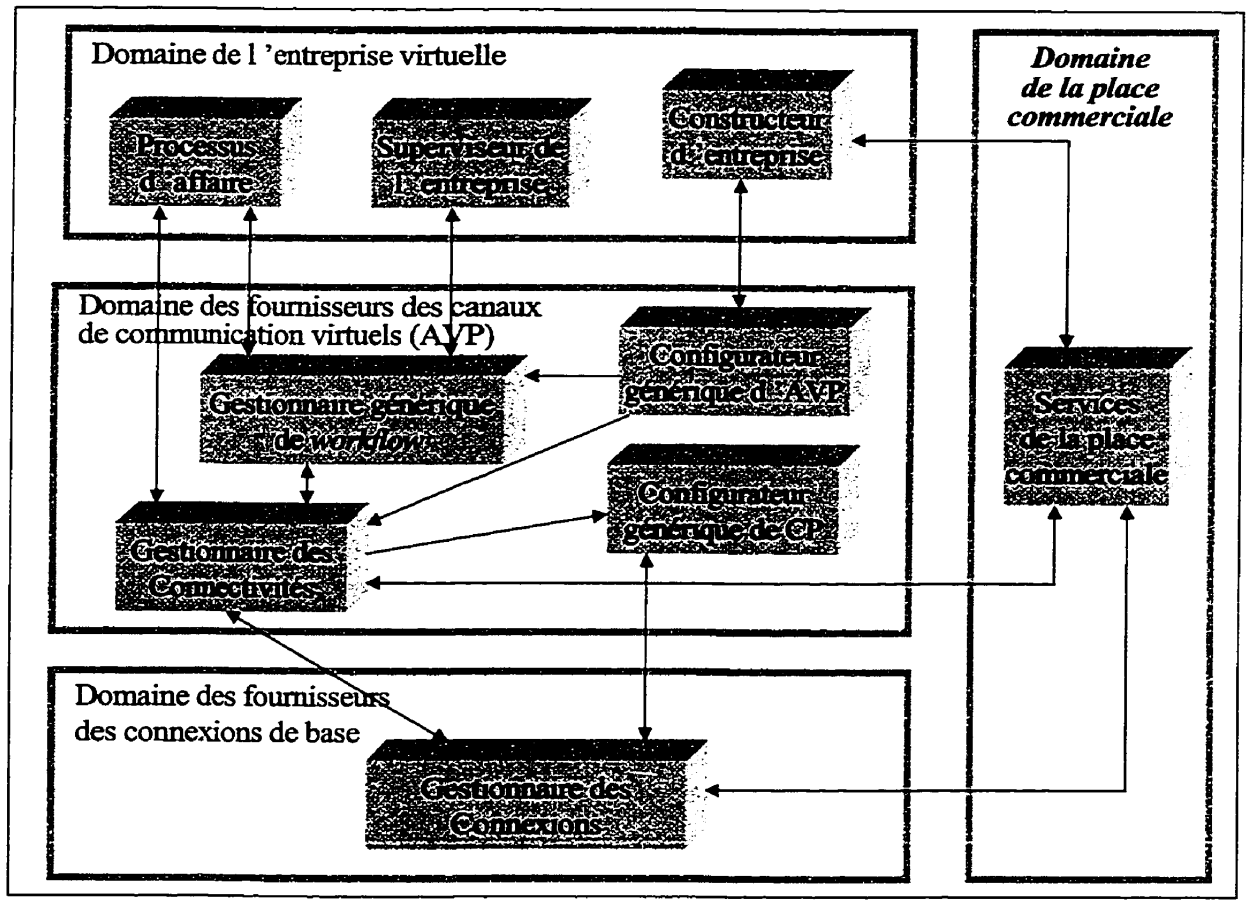


Figure 1.1: Architecture de la plate-forme MIAMI

Dans la plate-forme MIAMI, nous avons à gérer trois domaines d'activité : les fournisseurs de services de connexion de base (*Connectivity Providers — CP*), les fournisseurs de services de haut niveau (téléconférence, Internet, ...etc) (*Active Virtual Pipe Providers — AVP*) et les entreprises virtuelles (*Virtual Enterprise — VE*) qui utilisent aussi bien des services de base ou de haut niveau pour produire et commercialiser des services spécialisés (accès à des banques de données, formation à distance, commerce électronique,...etc.)

a- Domaine des fournisseurs de connexions de base : ce domaine comprend un ou plusieurs réseaux de transport appartenant à différents fournisseurs. Chaque fournisseur

annonce dans la place commerciale ses ressources disponibles, et attend de recevoir des offres pour entamer une négociation. Le module de gestion de connexion assure la configuration des ressources et la qualité de service selon le contrat établi avec le client.

b- Domaine des fournisseurs de canaux virtuels : le concept de canal de communication virtuel a été introduit par le consortium[8] MIAMI pour distinguer les services de haut niveau des connexions de base. Le concept de canal de communication virtuel (*Active Virtual Pipe — AVP*) fournit une vue abstraite d'un service global dynamique et auto-configurable pour la transmission de données. En fait, un service de canal de communication virtuel fournit un lien extranet dynamique et programmable en fonction de la demande du client. Puisque le fournisseur d'AVP n'est pas nécessairement propriétaire de réseaux de transport, il doit alors négocier avec un ou plusieurs fournisseurs de connexions pour acheter les ressources nécessaires à l'établissement de ses services de haut niveau. Ainsi, les agents de ce domaine utilisent la place commerciale pour l'élaboration de leurs contrats de vente ou d'achat. Le module '*configurateur générique d'AVP*' assure la configuration dynamique du canal de communication, les modules '*gestionnaire de connexion*' et '*configurateur générique de CP*' assurent le contrôle et la gestion des ressources de connexions, et le module '*gestionnaire générique de workflow*' assure la création et le contrôle du canal de communication et fournit une interface d'interaction avec l'entreprise virtuelle qui exploite le canal de communication.

c- Domaine d'entreprise virtuelle : le consortium MIAMI définit l'entreprise virtuelle comme étant une fédération temporaire de fournisseurs de services qui collaborent pour la réalisation d'un but commun. Les principales composantes du domaine d'entreprise virtuelle sont : le *'constructeur d'entreprise'* (*VEB*), le *'superviseur de l'entreprise'* et le *'processus d'affaire'*. Un agent qui voit une opportunité d'affaire, appelle le constructeur d'entreprise pour l'assister à la création de l'entreprise. En fait, le VEB consulte la place

---

[8] Plusieurs partenaires collaborent dans le cadre du consortium MIAMI: *GMD FOKUS Germany, Alcatel Bell Belgium, Alcatel CIT SA France, University College London United Kingdom, Imperial College of Science Technology and Medicine United Kingdom, The University of Surrey United Kingdom, Algosystems SA Greece, Centre de Recherche Informatique de Montreal Canada, France Telecom CNET France, Hitachi Ltd Japan.*

commerciale pour trouver des partenaires potentiels et négocier les contrats pour l'acquisition des ressources nécessaires à l'entreprise. Le '*superviseur de l'entreprise*' assure la gestion de celle-ci tout le long de son cycle de vie. Puisque l'entreprise virtuelle est constituée d'une fédération de partenaires autonomes, le superviseur doit posséder un processus de gestion distribué.

d- Domaine de la place commerciale : tous les agents représentant les différents domaines utilisent la place commerciale pour s'échanger des offres et demandes. Les agents VEB consultent la place pour chercher les ressources nécessaires à l'entreprise ou annoncer les services de celle-ci. Les agents AVP consultent la place pour acheter les services de connexion et annoncer leurs services de haut niveau. Les agents CP annoncent leurs ressources disponibles dans la place commerciale et les utilisateurs externes consultent la place commerciale pour acheter les services fournis par les entreprises virtuelles.

Dans le cadre de cette thèse, nous nous intéressons à la mise en œuvre du domaine de la place commerciale et des mécanismes de négociation qui assurent la coopération des agents représentant les différents domaines.

La plate-forme MIAMI constitue donc un système complexe pour la gestion des services de télécommunication et de commerce électronique, où plusieurs types d'agents doivent négocier pour satisfaire leurs buts[9]. En conséquence, l'objectif du modèle de négociation n'est pas seulement de permettre aux agents de maximiser leurs profits ou de minimiser leurs coûts, mais le nombre de buts satisfaits à chacun des trois niveaux de services est aussi d'une grande importance, puisque tous les contrats sur les domaines CP et AVP et VEB sont interdépendants. En fait, notre système n'est ni du type 'résolveur distribué de problème'[10] (*Distributed Problem Solver — DPS*) [Burckert 1991, Edmonds 1994] où seule la performance sociale est importante ni du type agents autonomes[11] [Fisher 1996, Durfee 1994] où seule la performance individuelle est importante. Nous sommes plutôt

---

[9] Dans ce contexte un but consiste à vendre ou acheter un service en fonction de certaines caractéristiques.

[10] Dans les systèmes DPS, le seul souci du concepteur est la performance globale du système et la rationalité individuel de chaque agent n'est pas importante.

[11] Dans les systèmes constitués d'agents autonomes, le concepteur qui généralement n'est pas le même pour tous les agents est concerné seulement par la performance individuel de son agent.

dans un contexte hybride où nous voulons exploiter la puissance conceptuelle des agents autonomes (comme dans les systèmes avec des agents autonomes), tout en assurant que le système global fonctionne de manière cohérente (comme dans le cas des systèmes DPS). Étant donné ces deux exigences contradictoires, nous croyons[12] que la meilleure façon de procéder pour construire un système multiagent socialement cohérent [Esmahi 1999d, Esmahi 1999e] est de doter les agents qui le composent d'une plus grande conscience au sujet de leurs interdépendances. Une telle conscience permet aux agents de proposer des offres et construire des contrats qui exploitent les relations de dépendances pour satisfaire un grand nombre de buts. Le but principal de ce travail est de vérifier cette hypothèse qui dit : "*l'intégration des relations de dépendances dans le processus de négociation va entraîner un équilibre entre le besoin de la rationalité individuelle*[13] *des agents et le besoin du fonctionnement cohérent du système global*"

1.3 Objectif scientifique et contributions

Ce travail se place ainsi dans le domaine de l'intelligence artificielle distribuée IAD [Demazeau 1989], plus particulièrement dans un de ses sous-domaines appelés systèmes multiagent (SMA) [Bond 1988]. Dans la section suivante nous allons préciser ce que nous entendons par agent, système multiagent, ainsi que d'autres termes liés au domaine. Cependant, de façon préliminaire, on peut simuler un SMA en un ensemble d'entités (agents logiciels ou humains) qui interagissent entre eux pour l'accomplissement de certains buts. Par conséquent, comme l'ensemble d'agents appartenant au groupe et la planification de buts ne peuvent pas être connus a priori, l'organisation du SMA doit être établie de manière dynamique. Ainsi, les agents forment dynamiquement des coalitions lorsqu'ils ne peuvent pas atteindre leurs buts tout seuls. Dans ce contexte, la possibilité de raisonner sur les autres, et d'évaluer le potentiel des autres agents à coopérer sont des fonctionnalités essentielles chez chaque agent, puisque les agents ne sont pas censés a priori s'aider les uns les autres.

---

[12] Le verbe 'croire' est utilisé ici dans le sens de faire une hypothèse.

[13] Un agent est dit individuellement rationnel s'il n'accepte que des contrats qui lui permettent de réaliser un profit positif.

Le principal objectif de ce travail est de faire une étude exploratoire sur les possibles avantages d'une approche fondée sur la complémentarité entre la notion d'utilité et les dépendances bilatérales entre les agents pour la construction d'un modèle de coopération dans un SMA.

Par rapport à notre objectif, notre intention est d'utiliser les relations de dépendances bilatérales en plus des fonctions d'utilité, comme un critère de mesure de la possibilité qu'un agent possède pour adopter un but de l'autre, comme présenté dans (cf. 5.3.2.). Ainsi, en utilisant la théorie de la dépendance [Castelfranchi 1992] et la théorie de l'utilité [Ponsard 1977], notre modèle de coopération (cf. 5.) fournit un cadre où l'agent peut raisonner sur les buts des autres pour proposer la formation d'une coalition à des agents qu'il croit les plus réceptifs, et si ceux-ci décident de ne pas vouloir en faire partie par manque d'intérêt, il peut utiliser un mécanisme de persuasion pour les convaincre.

Par ce travail nous montrons qu'un tel modèle fournit les contributions suivantes dans le domaine des SMA. :

- d'un point de vue **modélisation de l'interaction** dans les SMA, un tel modèle nous permet de prendre en compte la notion d'adoption de buts, chose absente dans les approches fondées exclusivement sur la notion d'utilité,
- en ce qui concerne **l'adaptation des agents**, les graphes de dépendances permettent à chaque instant à l'agent de savoir parmi ses buts lesquels sont réalisables en fonction de la dynamique des agents dans la société,
- en ce qui concerne **la formation de coalitions**, notre modèle introduit d'une part, la notion de persuasion pour permettre à un agent de convaincre, par argumentation, ses pairs à adopter ses buts et d'autre part, la notion d'engagement social pour faire un équilibre entre la rationalité individuelle et la cohérence sociale dans le système,
- en ce qui concerne **la révision de croyances**, le modèle fournit un mécanisme exploitant les graphes de dépendances pour détecter s'il existe une inconsistance entre les représentations que les agents ont les uns sur les autres,
- du côté **implantation**, l'architecture définie pour l'implantation du modèle sur la plate-forme MIAMI, fournit une contribution intéressante pour le domaine du fait que

MIAMI est une plate-forme de SMA où l'interaction et la négociation entre agents est d'un grand intérêt.

1.4 Organisation de la thèse

Ce manuscrit est composé de huit chapitres organisés de la manière suivante :

- le chapitre 1, décrit nos motivations scientifiques, définit le contexte de réalisation de ce travail, introduit la problématique abordée, précise l'objectif du travail et décrit l'organisation du manuscrit;
- le chapitre 2 spécifie le domaine de recherche de cette thèse, en présentant entre autres un bref historique et les principaux axes de recherche (IAD, SMA). Les concepts reliés à ces deux axes (agent et coopération) sont aussi définis;
- le chapitre 3 présente une analyse de la problématique de coopération dans les SMA. Par une analyse de l'interaction entre agents dans les SMA, nous avons identifié les principales variantes qui influencent le comportement des agents dans de tels systèmes et nous avons mis en relief l'importance des relations de dépendances dans cette interaction;
- le chapitre 4 fait un tour d'horizon des principales approches de recherche sur la modélisation des mécanismes de négociation dans les deux grands domaines de recherche sur ce sujet : la théorie des jeux et les systèmes multiagent. Nous relevons quelques problèmes intéressants avec ces approches et les idées que nous avons retenues pour notre approche;
- le chapitre 5 présente le modèle général de coopération que nous avons construit dans le cadre de ce travail. Nous présentons les principes utilisés pour l'élaboration de ce modèle, nous décrivons les procédures de négociations sous-jacentes et nous décrivons également une stratégie de négociation que nous avons utilisé pour implanter nos agents courtiers dans MIAMI ;
- le chapitre 6 décrit le prototype MIAMAP (*MIAMI Market Place*), pour implanter le domaine de la place commerciale dans MIAMI qui exploite une bonne partie du modèle que nous avons défini dans le chapitre 5. L'architecture de la place

commerciale, ainsi que l'implantation des différents agents qui interagissent dans MIAMAP sont détaillées ;

- le chapitre 7 présente une analyse des résultats de simulation du prototype implanté;
- finalement, nous concluons la thèse en établissant les points forts et les limites de notre modèle, ainsi que les voies de recherche qui nous paraissent les plus prometteuses pour le futur.

Chapitre 2: Présentation du domaine de recherche

2.1 Bref historique

Au milieu des années 70 et début des années 80, avec le développement des technologies de réseaux, il est devenu techniquement possible et économiquement avantageux de remplacer les gros ordinateurs par des réseaux d'ordinateurs personnels ou des stations de travail. Au même moment, les premiers modèles [Lenat 1988, Agha 1988] ou systèmes [Bisiani 1987] permettant l'adoption d'une approche distribuée de l'IA sont apparus. C'est la naissance d'un nouveau domaine de recherche, l'intelligence artificielle distribuée (IAD) et, conséquemment, d'un de ses sous-domaines, la résolution distribuée de problèmes (RDP).

La caractéristique principale de cette première génération de systèmes d'IAD est la distribution des traitements qui ne sont plus activés de façon séquentielle, mais exécutés de façon concurrente. Cependant, ces systèmes continuent à être fermés et leurs traitements continuent à être non réutilisables. L'objectif principal était de résoudre de façon distribuée un problème bien déterminé. Les techniques d'interaction ou de coopération des éléments du système sont développées, mais généralement demeurent très dépendantes du modèle algorithmique sous-jacent adopté pour le problème en question. En fait, ils utilisent l'approche classique basée sur le modèle du tableau noir.

Plus tard, pendant les années 80 et au début des années 90, des efforts sont menés dans le sens de permettre une réutilisabilité des traitements. Pour cela, diverses méthodes pour implanter un contrôle décentralisé sont proposées et testées [Demazeau 1989]. Les activités de conception des sous-systèmes, les moyens pour les faire communiquer et les moyens pour concevoir leur contrôle et leur coordination deviennent disjoints. Ainsi l'IAD donne naissance à un autre sous-domaine : les systèmes multiagent (SMA).

L'importance de représenter explicitement au sein de chaque agent les informations sur les autres agents, dans un contexte de planification, est discutée par Konolige [Konolige

1980]. L'utilisation des modèles des organisations humaines dans les systèmes afin de résoudre un problème de façon coopérative est proposée par Fox [Fox 1981]. En ce qui concerne l'allocation de tâches, le protocole de négociation Contract-Net est présenté par Smith [Smith 1988a]. Finalement, les premiers environnements informatiques pour tester ces idées apparaissent, tel : DVMT [Durfee 1987], MACE [Gasser 1987], ARCHON [Jennings 1992, 1993].

2.2 Paradigme agent

Le paradigme agent est issu des approches de l'intelligence artificielle (IA). Depuis plusieurs décennies, de nombreux chercheurs comme V. Lesser [Lesser 1975], K. Konolige [Konolige 1980], M. Minsky [Minsky 1985], et d'autres ont étudié divers problèmes dans les domaines de l'IA, de la robotique et des technologies connexes, visant à améliorer les performances humaines et mettent en jeu diverses formes de technologie agent. Un grand nombre d'expressions et de termes différents sont utilisés par les chercheurs pour décrire les agents sur lesquels ils travaillent, comme: agents intelligents, interfaces intelligentes, knowbots pour '*knowledge robots*', softbots pour '*software robots*', userbots, taskbots, agent autonome, agent mobile ou agent de communication.

En fait, le mot agent a la caractéristique d'être très général. Il convient aussi bien à un humain qu'à un robot ou à un programme informatique.

A. Kay [Kay 1990] définit l'agent comme un processus intelligent en arrière plan qui peut fructueusement cloner les objectifs de son utilisateur et les accomplir. Quant à T. Selker [Selker 1994], il définit l'agent comme un programme informatique qui simule une relation humaine, en faisant des tâches à son utilisateur qu'une autre personne peut lui faire. M. Minsky [Reicken 1994] rapporte, dans une interview, "une machine qui accomplit quelque chose, sans avoir besoin de comprendre son fonctionnement, vous l'appelez un agent quand vous voulez la traiter comme une boite noire". M. P. Singh [Singh 1994] définit l'agent comme étant un système intelligent qui permet de capter l'attitude intentionnelle. N. R. Jennings et M. Wooldridge [Jennings 1998] définissent

l'agent comme étant un système informatique situé dans un environnement et qui est capable de faire des actions de façon autonome afin de satisfaire l'objectif pour lequel il a été conçu. Alors que S. J. Russel [Russel 1995] le définit comme étant une entité qui peut percevoir son environnement à l'aide de détecteurs et agit de façon autonome et rationnelle sur cet environnement par ses actuateurs. Par ailleurs, Y. Shoham [Shoham 1993] parle de l'agent comme une entité dont l'état est vu comme étant constitué de composantes mentales, telles que, croyances, capacités, décisions et engagement, ... puis ajoute; "ce qui fait d'une composante matérielle ou logiciel un agent, c'est précisément le fait qu'on choisisse de l'analyser et la contrôler selon ces concepts".

En fait, les auteurs ne s'entendent pas sur une définition précise de ce que c'est un agent [Franklin 1996]. Cependant, ils définissent un ensemble de caractéristiques souhaitables pour un agent:

- *l'autonomie*: un agent prend l'initiative et exerce un contrôle sur ses actions selon différentes manières: (1) orienté but[14]: un agent accepte des requêtes de haut niveau et est capable de satisfaire leur but, (2) coopératif: il n'obéit pas aveuglément aux commandes. Il peut demander une clarification, modifier la requête ou même refuser de la satisfaire, (3) flexible: il peut dynamiquement choisir quelles actions appliquer et dans quel ordre, en fonction de l'état de son environnement, (4) auto-déclenchable: un agent peut détecter un changement dans son environnement et décider d'agir,
- *l'adaptabilité*: un agent s'adapte automatiquement aux préférences de son utilisateur ou à son environnement,
- *la continuité temporelle*: un agent est un processus continu et non pas un programme qui s'exécute en traitant des entrées pour produire des résultats et s'arrêter,
- *la personnalité*: un agent possède des attitudes intentionnelles qui facilitent son interaction avec les autres agents,

[14] Plusieurs systèmes multiagents sont plutôt orientés tâche que but. Ainsi dans ce contexte, quand le terme but est utilisé on peut toujours penser à une tâche ou un ensemble de tâches interdépendantes.

- *la capacité de communication*: un agent est capable de s'engager dans une communication complexe avec d'autres agents, y compris l'humain, pour obtenir des informations ou demander de l'aide afin d'accomplir un but,
- *la sociabilité*: un agent est capable de coopérer avec d'autres agents pour satisfaire des buts communs,
- *la rationalité*: dans le sens de la théorie d'utilité, un agent agit de façon à maximiser sa performance en fonction d'une évaluation d'utilité,
- *la mobilité*: un agent peut migrer vers différents sites de réseaux même hétérogènes dans le but d'accomplir progressivement les tâches qui lui sont confiées.

Dans ce contexte nous considérons la structure de l'agent comme étant composée d'au moins quatre parties essentielles :

- *composante réactive*: permettant d'implanter le comportement réactif de l'agent (tâches de communication, actions ne nécessitant aucune activité de raisonnement ou de planification),
- *état mental*: composée de croyances, capacités et engagements. Les croyances représentent l'ensemble des faits quantifiés temporellement [Bestougeff 1989] que l'agent croit être vrai ou faux. Les capacités représentent l'ensemble des tâches que l'agent peut accomplir. Les engagements représentent l'ensemble des tâches ou actions quantifiées temporellement pour lesquelles l'agent a pris un engagement envers ses pairs ou qu'il a planifié pour son propre compte,
- *composante de contrôle*: permettant d'implanter le comportement cognitif de l'agent. Elle comprend tous les mécanismes de raisonnement et les règles de contrôle,
- *composante sociale*: permettant d'implanter les mécanismes de coordination et les facilités de négociation chez l'agent. Elle implante aussi toutes les normes ou lois régissant le comportement social dans le système.

2.3 Systèmes multiagent versus résolution distribuée de problèmes

Comme nous l'avons déjà discuté dans la section 2.1., soit pour des raisons historiques, soit par des motivations scientifiques diverses, le domaine de l'IAD peut être divisé en deux sous domaines : la RDP et les SMA.

Le point de départ de l'approche RDP est un problème précis devant être résolu. Ainsi, du point de vue de conception du système, les agents n'existent pas à priori. La définition de leur conception, ainsi que de leur organisation et de leurs interactions est due à l'existence d'un problème que le système doit résoudre. La réutilisabilité des agents, de leurs interactions et de leur organisation n'est donc pas possible.

La problématique abordée par les SMA est très différente de celle de la RDP. En fait, les SMA abordent l'étude de l'activité des agents autonomes dans un univers multiagent. Comme il n'existe pas à priori de problème que le système est sensé résoudre, il s'agit d'étudier des modèles généraux à partir desquels nous pouvons concevoir les agents, les organisations et les interactions, de façon à pouvoir les instancier lorsqu'on doit résoudre un problème particulier. Autrement dit, on conçoit les moyens grâce auxquels on peut assurer que les agents vont coopérer les uns avec les autres pour résoudre un certain problème, quand celui-ci sera posé au système. Il devient donc possible de réutiliser ces composantes pour concevoir des applications similaires.

2.4 Autonomie versus sociabilité

Dans ce paragraphe, nous n'essayons pas de définir les notions d'autonomie [Castelfranchi 1995] et de sociabilité [Castelfranchi 1990], mais plutôt de formuler quelques postulats cruciaux sur lesquels nous nous basons pour définir le modèle de coopération à base des relations de dépendances (cf. 5.). Ainsi, dans ce contexte nous considérons que l'autonomie est un concept relationnel et nécessairement limitée. En effet, l'interprétation « *relationnel* » vient du fait que nous considérons l'autonomie comme un concept social. L'agent est autonome par rapport à une relation d'influence avec d'autres agents, et l'autonomie des agents est limitée à cause de cette sociabilité,

autrement l'autonomie sera contradictoire au contexte de SMA et à l'avantage de la sociabilité.

Ainsi, la notion d'autonomie sociale [Castelfranchi 1998] peut être définie pour un agent par les postulats suivants :

1. l'agent possède ses propres buts internes, qui ne dérivent pas de la volonté d'autres,
2. l'agent est capable de prendre des décisions concernant des buts en conflit (que ce soit ses propres buts ou des buts qu'il a adoptés de l'extérieur),
3. l'agent peut adopter des buts d'autrui et il est assujetti à l'influence. En fait, il peut être convaincu par un autre agent d'adopter un but,
4. l'agent adopte les buts des autres en conséquence d'un choix sur tous les buts qu'il a en mains à un moment donné. Ainsi l'adoption n'est ni automatique ni dirigée par des règles,
5. l'agent adopte les buts des autres de façon rationnelle, c'est à dire, seulement s'il voit que l'adoption est un moyen lui permettant de réaliser certains de ses buts. Cela n'est possible que si les agents sont interdépendants sur leurs buts et ainsi possèdent une force d'influence les uns sur les autres. Cependant, leur autonomie est assurée par le fait qu'ils décident à base de leurs buts et croyances d'adopter ou non les demandes de leurs collègues,
6. il n'est pas possible de modifier directement les buts de l'agent; toute modification dans ses buts doit passer par la modification de ses croyances. Ainsi, le contrôle sur les croyances devient un filtre de contrôle sur l'adoption de buts,
7. il est impossible de changer automatiquement les croyances de l'agent. L'adoption de croyances est une décision que l'agent prend en fonction de plusieurs critères et contrôles. Ainsi l'autonomie cognitive de l'agent est protégée.

Les postulats 6 et 7 définissent ce qu'on appelle un double filtre dans la structure d'autonomie des agents sociaux.

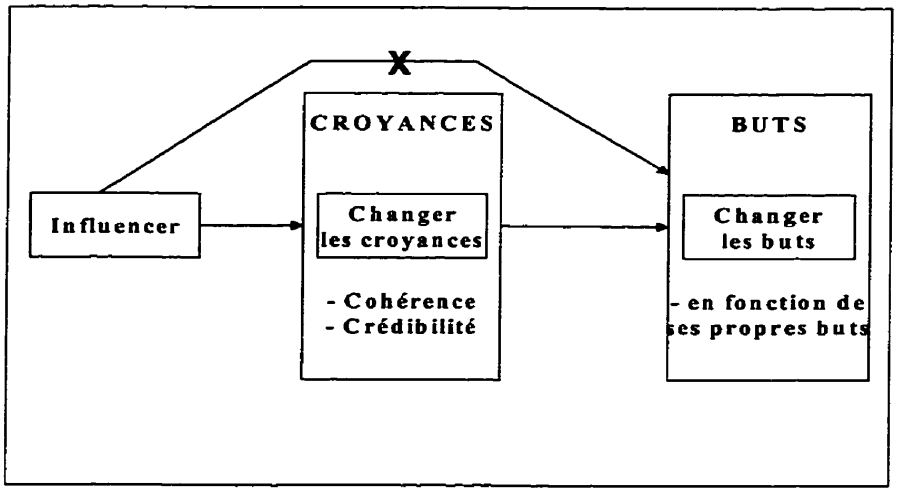


Figure 2.1: Double filtre pour le contrôle des buts

Un agent autonome contrôle l'acquisition de nouvelles croyances en testant sur (1) la cohérence avec les croyances qu'il possède (pour garder la consistance de sa base de croyances) et (2) la crédibilité et les compétences de la source. Pour le premier critère, un agent accepte une croyance s'il peut la justifier ou du moins l'insérer sans inconsistance dans sa base, et pour le deuxième critère, l'agent accepte les informations fournies par un agent sur lui-même. Ces deux critères entraînent une propriété intéressante des croyances. En fait, un agent accepte des croyances selon leur cohérence avec les connaissances qu'il possède ou selon la logique des arguments qui les supportent, et non pas selon son intérêt ou utilité.

En conséquence, pour persuader un agent à faire quelque chose, il faut absolument l'amener à croire à ce quelque chose. Ce mode de révision des croyances assure donc une protection générale à l'autonomie des buts (double filtre, figure 2.1).

2.5 Coopération entre agents

Les chercheurs se sont intéressés au problème de la coopération d'agents intelligents dans deux principaux domaines : (1) la théorie des jeux et (2) l'intelligence artificielle distribuée avec ses deux sous-domaines, systèmes multiagent et systèmes de résolution distribuée de problèmes. Les chercheurs du domaine de la théorie des jeux [Pruitt 1981] se sont intéressés à construire des modèles par lesquels un agent autonome peut décider

de coopérer dans une situation donnée. Les chercheurs du domaine des systèmes multiagent [Georgeff 1983, Tennenholtz 1989] croient qu'une coopération surgit lorsque deux agents ayant des buts à satisfaire se rencontrent et déterminent qu'ils peuvent unir leurs ressources pour réaliser des tâches relatives à leurs buts. Ainsi, la coopération dans ce contexte n'est ni automatique ni irrationnelle. Quant aux chercheurs du domaine de la résolution distribuée de problèmes [Cammarata 1983, Durfee 1989], ils croient que les agents doivent être construits dans l'intention de coopérer les uns avec les autres. En fait, dans ce contexte la coopération est basée sur une organisation des agents définie par le concepteur.

Dans notre contexte (SMA), nous supposons que les agents forment des coalitions quand ils croient qu'il est nécessaire de travailler de façon coopérative pour atteindre un certain but. Nous nous inspirons du modèle formel de résolution coopérative de problèmes proposé par Wooldridge et Jennings [Wooldridge 1994] et qui comprend les quatre phases représentées dans la figure suivante:

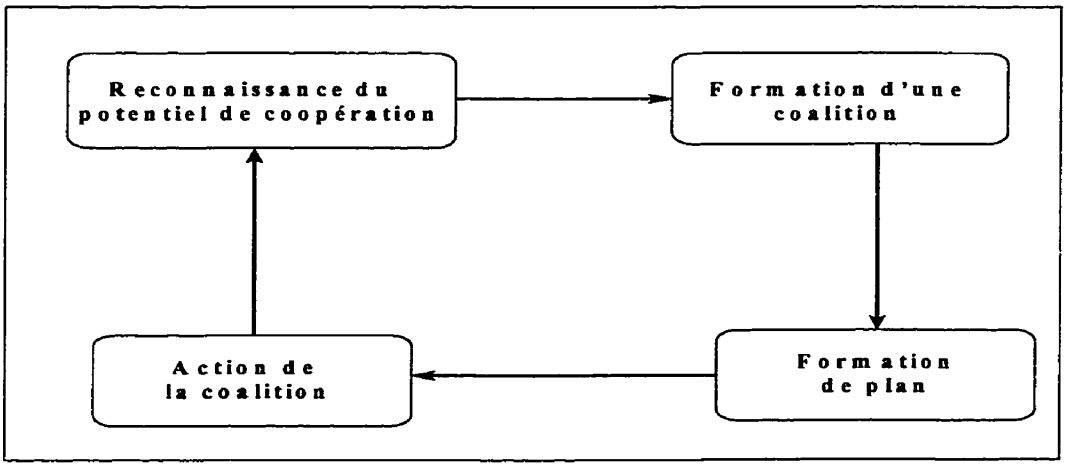


Figure 2.2: Phases de coopération

a- Reconnaissance du potentiel de coopération: une résolution coopérative de problème commence quand un agent reconnaît le potentiel pour une action coopérative. Cette reconnaissance apparaît quand l'agent, ayant un but à atteindre, ne peut pas l'atteindre seul ou bien préfère l'atteindre en coopération avec d'autres agents,

b- Formation d'une coalition: pendant cette phase, l'agent qui a reconnu le potentiel de coopération demande de l'aide aux autres en communiquant avec eux. Si cette phase est bien réussie, il y aura formation d'une coalition, c'est à dire formation d'un groupe d'agents ayant un engagement commun pour une certaine action collective,

c- Formation de plan: pendant cette phase, les agents essaient de négocier un plan commun pour atteindre le but désiré. Un mécanisme de négociation permet aux agents de montrer leurs préférences par rapport au plan devant être choisi,

d- Action de la coalition: après avoir été choisi, le plan est exécuté par des agents qui maintiennent une relation très stricte entre eux. Des conventions sont suivies pour assurer une coordination des diverses actions et pour détecter la fin de l'activité commune (par exemple lorsqu'un agent détecte que le but a été atteint).

Une fois l'activité commune terminée, la coalition disparaît. Bien sûr, le cycle peut être exécuté à nouveau.

Chapitre 3: Analyse de la problématique de coopération dans les systèmes multiagent

3.1 Présentation du problème

Dans le cadre de la problématique introduite dans la section 1.2, nous nous plaçons dans un SMA ouvert, où les agents peuvent entrer ou sortir dynamiquement. Par conséquent, la formation de coalition se fait aussi de façon dynamique.

Deux questions fondamentales se posent alors pour la coopération dans de tels SMA: (1) le problème de sociabilité, c'est à dire, quel mécanisme de motivation pousse l'agent à accepter d'entrer dans une activité sociale, et (2) le problème d'adoption, c'est à dire, comment un agent peut rendre son but social. Dans ce travail, nous présentons un modèle de coopération où la réponse à la première question est basée sur *la théorie de l'utilité* et la réponse à la seconde question est basée sur *la logique de persuasion.*

Dans ce contexte, la possibilité de raisonner sur les autres est une fonctionnalité essentielle pour l'agent « intelligent ». En particulier, un agent doit pouvoir évaluer à tout moment si ses buts sont réalisables[15] et si ses plans sont exécutables[16]. De plus, lorsqu'un agent ne peut atteindre ses buts tout seul, il doit pouvoir évaluer la possibilité des autres agents à coopérer avec lui et les convaincre, puisque les agents ne sont pas censés, à priori, s'aider les uns les autres. Ainsi, en raisonnant sur les buts des autres agents, un agent peut proposer la formation d'une coalition à ceux qu'il croit être les plus réceptifs à une telle proposition et ceux-ci peuvent décider s'il est profitable ou non pour eux d'en faire partie.

En plus, un agent doit pouvoir réviser ses croyances relatives aux autres agents au fur et à mesure qu'il interagit avec eux car, dans un tel système, il est peu probable que les agents aient à tout moment des informations correctes et complètes les uns sur les autres. Aussi,

---

[15] Un but est dit réalisable s'il existe un plan exécutable qui peut l'atteindre.

[16] Un plan est dit exécutable si toutes les fonctionnalités nécessaires pour l'accomplir sont présentes.

la participation à une coalition est individuellement rationnelle pour chaque agent, c'est à dire, l'agent ne coopère que si la solution lui est profitable. Finalement, puisque les ressources des agents sont limitées, les mécanismes de coopération doivent prendre en considération ces contraintes. Conséquemment, notre travail se classe parmi ceux qui défendent la thèse qu'un agent doit posséder un mécanisme de raisonnement social pour réagir correctement dans ce genre de situation, par opposition à la thèse que le comportement social peut émerger d'un échange réactif de messages sans que l'agent n'utilise une représentation des modèles de ses pairs.

Dans notre contexte, l'existence d'un tel mécanisme de raisonnement social s'appuie sur les prémisses suivantes :

- un agent doit posséder une représentation explicite ou avoir accès à un modèle pour chacun de ses pairs. Un tel modèle a la caractéristique lui aussi d'être mis à jour dynamiquement. Chaque agent a sa propre représentation des autres, qui lui est privée. L'acquisition de cette représentation peut être réalisée par différentes sources : la perception, la communication ou l'inférence,
- un agent doit exploiter ses modèles ainsi que toutes les relations de dépendances déduites de ses modèles pour optimiser son comportement vis-à-vis de ses pairs. Cette exploitation, basée sur l'évaluation de son profit local lui permet de détecter si, à un moment donné, ses buts sont réalisables ou non et ses plans son exécutables ou non, d'évaluer la possibilité des autres agents d'accepter de former une coalition pour atteindre ses buts, et de décider s'il est convenable d'accepter de faire partie d'une telle coalition,
- un agent doit réviser ses modèles lorsqu'il détecte que les informations qu'il possède sur les autres sont incomplètes ou incorrectes. Cette révision est réalisée de façon autonome, c'est à dire, sans contrôle global préétabli,
- la coopération prend place dans un monde dynamique, et pendant la négociation les conditions qui affectent le comportement et les buts des agents peuvent changer. Les prévisions des agents et leurs attentes sur le comportement des autres peuvent s'avérer incorrectes. Ainsi, l'agent a besoin de prendre en considération tout

changement dans son environnement. En d'autres termes, il doit avoir une composante à comportement réactif,

- puisque l'entente finale est atteinte en réduisant la différence entre les demandes des deux parties, l'agent a besoin d'une méthode de prévision et d'évaluation pour s'assurer qu'une solution réduit bien cette différence,
- afin d'atteindre une entente, chaque agent doit modifier partiellement ou totalement les plans d'accomplissement de certains de ses buts. Ainsi, l'agent doit avoir une composante pour générer de l'argumentation pour convaincre ses pairs. Nous avons développé un mécanisme de ce type fondé sur la notion d'argumentation et de persuasion [Esmahi 1999a] et ce, en utilisant le raisonnement par cas, la théorie d'utilité et les graphes de dépendances des buts des agents.

3.2 Analyse de la problématique d'interaction dans les SMA

L'interaction dans les systèmes multiagent peut être analysée sur plusieurs plans :

- les relations sociales dans le système, déterminées par les types d'interactions entre les agents,
- les relations bilatérales entre les agents, déterminées par les relations de dépendances reliant les agents,
- l'état interne de l'agent lui-même, déterminé par la situation de l'interaction par rapport à l'état mental de l'agent.

Sur le plan social du SMA, les agents peuvent se trouver dans l'une des situations suivantes : interaction coopérative symétrique, interaction de compromis symétrique, interaction coopérative / de compromis non symétrique ou interaction de conflit. Sur le plan des relations bilatérales, les agents peuvent être dans l'une des situations suivantes : situation d'indépendance, situation de dépendance unilatérale ou situation de dépendance mutuelle. Finalement, en ce qui concerne l'état mental de l'agent, ce dernier peut se trouver dans l'une des trois situations face aux requêtes échangées pendant l'interaction : situation routinière, situation familière ou situation non familière.

Les paragraphes suivants présentent chacune des situations citées ci-dessus.

3.2.1 Types d'interactions dans un SMA

Sur le plan social, l'interaction des agents dans un SMA peut être de quatre types : interaction coopérative symétrique, interaction de compromis symétrique, interaction coopérative / de compromis non symétrique ou interaction de conflit. Pour établir une définition des différents types d'interaction, considérons l'environnement suivant : un SMA composé de deux agents A_1 et A_2, ayant des buts G_1 et G_2. Soit S_0, l'état initial du système, et E_1 et E_2 l'ensemble des états du système qui satisfont respectivement les buts de A_1 et A_2. L'intersection de E_1 et E_2 notée Φ, est l'ensemble des états pour lesquels un plan conjoint peut être négocié pour satisfaire simultanément les buts G_1 et G_2. En analysant l'interaction d'un point de vue individuel pour un agent, on distingue les quatre situations suivantes :

a- Interaction coopérative symétrique : dans une telle situation, les deux agents peuvent satisfaire leurs buts individuellement. Cependant, il existe un état de Φ qui est préférable (le plan conjoint est moins coûteux) pour les deux agents. Ainsi, les deux agents préfèrent coopérer que de satisfaire leurs buts de façon individuelle.

Une telle situation peut être visualisée par la figure suivante :

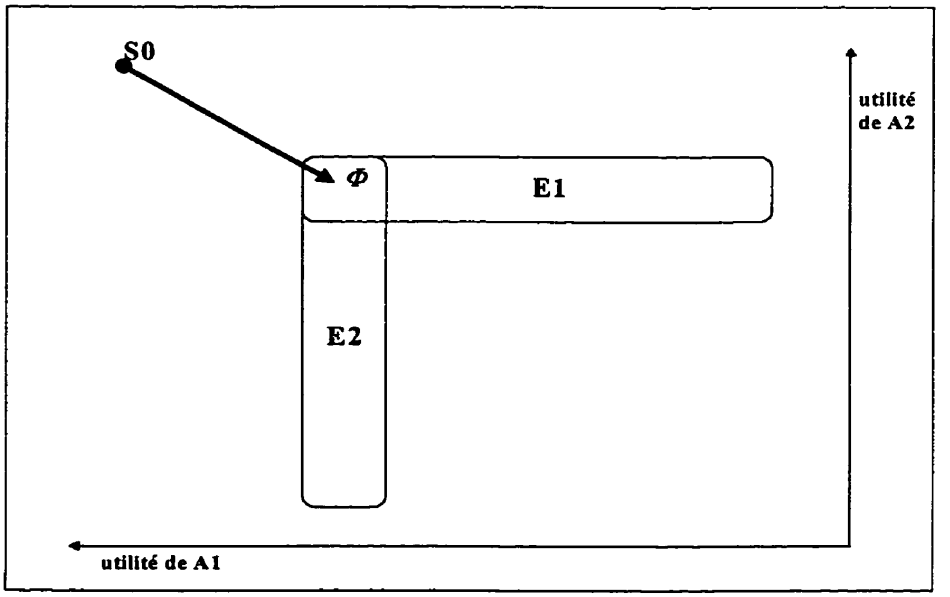


Figure 3.1: Situation coopérative symétrique

Un point dans le plan représente un état du système. Chaque rectangle représente l'ensemble des états qui satisfont le but d'un agent. L'intersection des deux rectangles E_1 et E_2 représente l'ensemble Φ. La flèche en gras représente le plan conjoint de A_1 et A_2 pour amener le système de son état initial S_0 à un état qui satisfait leurs buts. La distance entre S_0 et un point dans l'un des rectangles représente le coût associé au plan de l'agent qui permet d'amener le système de l'état S_0 à l'état représenté par le point considéré. Ainsi, les deux axes du graphe représentent le sens d'évolution de l'utilité de chacun des agents.

b- Interaction de compromis symétrique *:* dans une telle situation, il existe des plans plus profitables et individuellement rationnels pour les deux agents. Cependant, vu leur interaction dans le même environnement (i.e. Conflit sur les ressources), les agents ne peuvent pas utiliser leurs plans individuels et doivent mutuellement tenir compte de la présence de l'autre et négocier un plan conjoint pour satisfaire leurs buts. En fait, dans cette situation, un plan conjoint qui satisfait les deux buts existe mais les deux agents doivent faire une concession sur le profit à gagner. Toutefois, n'importe quel état dans leur ensemble de négociation Φ est meilleur que l'état initial S_0.

Une telle situation peut être visualisée par la figure suivante :

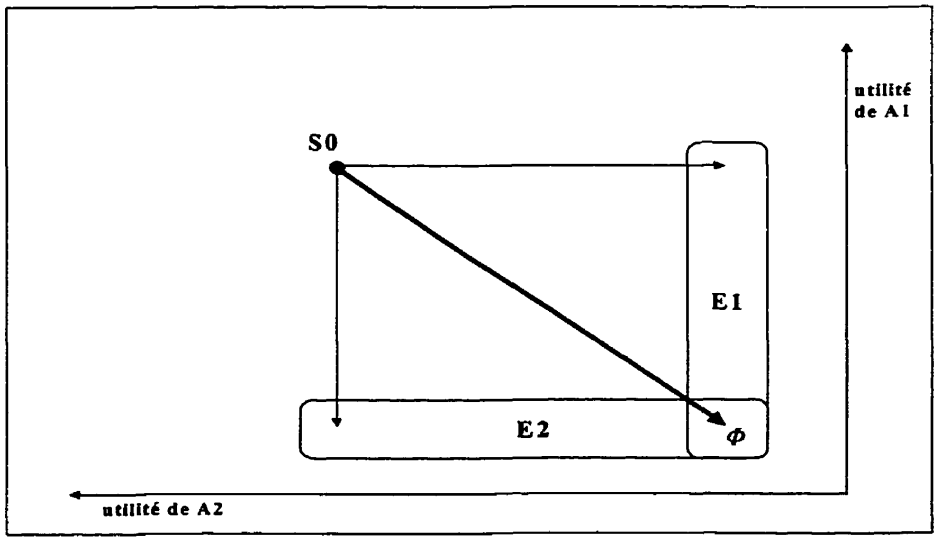


Figure 3.2: Situation de compromis symétrique

La flèche en gras représente le plan conjoint de A_1 et A_2 pour satisfaire leurs buts. Les flèches simples représentent respectivement les plans individuellement rationnels des deux agents.

c- Interaction coopérative / de compromis non symétrique : l'interaction dans une telle situation n'est pas symétrique entre les deux agents. En effet, un des agents voit l'interaction comme étant coopérative et préfère un plan conjoint à son plan individuel, par contre l'autre agent trouve l'interaction comme étant une situation de compromis puisqu'il préfère son plan individuel à tout plan conjoint. Toutefois, dans une telle situation, un plan conjoint reste toujours préférable pour les deux agents à laisser le système dans son état initial S_0. La figure suivante représente une telle situation :

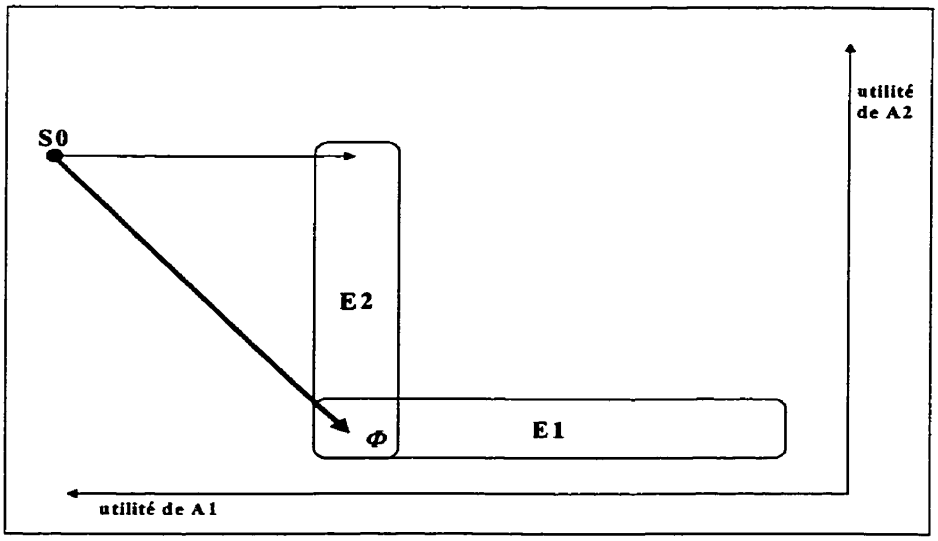


Figure 3.3: Situation coopérative / de compromis non symétrique

La flèche en gras représente le plan conjoint de A_1 et A_2 et la flèche simple représente le plan individuel préférable pour A_2.

d- Interaction de conflit : c'est une situation où l'ensemble Φ de négociation des agents est vide. En fait, il n'existe aucun plan conjoint acceptable pour les deux agents.

La figure suivante représente une telle situation dans l'espace des états d'interaction :

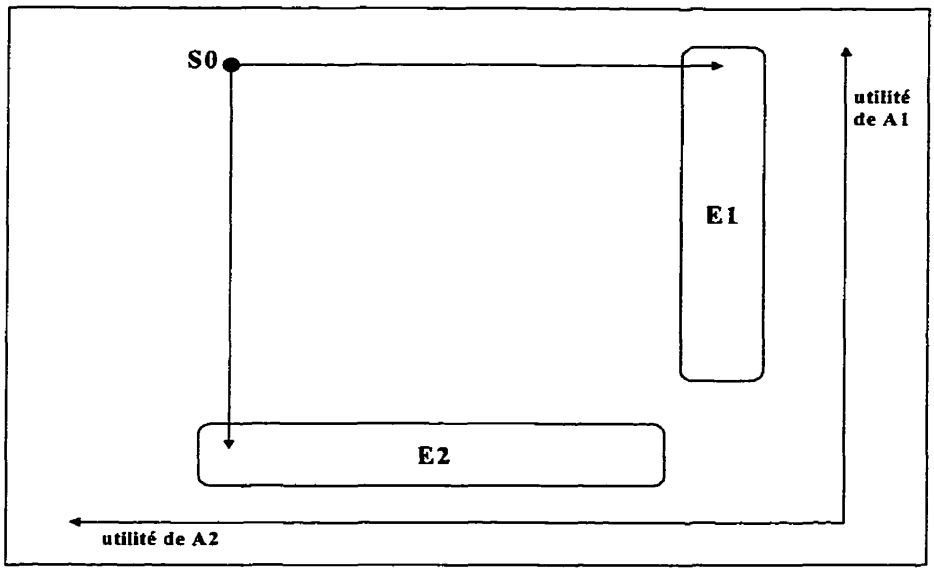


Figure 3.4: Situation de conflit

Aucun plan conjoint n'est possible puisque l'intersection des ensembles d'états acceptable pour les deux agents est vide. Les deux flèches simples représentent les plans individuellement rationnels pour chaque agent.

3.2.2 Situations observées en fonction de l'état mental de l'agent

L'analyse du comportement des agents selon les situations auxquelles ils font face dans l'environnement multiagent [Chaib-draa 1996] a permis de distinguer trois principales situations :

a- Situations routinières *:* se sont les situations auxquelles l'agent répond par un simple comportement réactif. Par ce comportement, l'agent est capable de traiter les changements rapides se produisant dans son environnement et l'information perçue suit l'un des deux chemins suivants : (1) *perception* → *action,* ou (2) *perception* → *identification* → *action.*

Généralement, le raisonnement utilisé par l'agent à ce niveau est sensé représenter la performance de l'agent durant des routines, c'est à dire des actes sans contrôle "conscient".

b- Situations familières *:* se sont les situations qui nécessitent un traitement de raisonnement et de planification explicite, toutefois l'agent possède les plans ou les

fonctions de raisonnements nécessaires pour faire face à la situation. L'information perçue dans l'interaction suit alors l'un des chemins suivants : (1) *perception → planification → action*, ou (2) *perception → identification → planification → action.*

c- Situations non familières : se sont les situations où l'agent est soit typiquement confronté à l'ambiguïté de différents choix, soit confronté à l'impossibilité d'identifier une situation donnée en termes d'action ou de but[17]. L'information provenant de l'interaction dans ce type de situation suit le chemin suivant : *Perception → identification, ou reconnaissance de cas similaire → adaptation → planification → action.*

3.2.3 Relations bilatérales entre les agents

Dans cette section, nous allons essayer d'analyser l'interaction entre les agents en terme des relations bilatérales reliant les agents.

Considérons les variables suivantes modélisant la situation d'interaction entre deux agents A_1 et A_2 :

C_i : le coût de réalisation du but G_i de A_i

P_i : le coût maximal acceptable par A_i pour la réalisation de son but G_i.

T : le coût total relatif au plan conjoint permettant la réalisation des buts de A_1 et A_2.

Mr_i : le coût relatif au rôle minimal que A_i peut jouer dans le plan conjoint.

a- Relation d'indépendance : les deux agents sont dits indépendants si chacun des agents possède un plan pour réaliser seul son but ($C_i < P_i$, $i = 1, 2$), alors qu'il n'existe pas de plan conjoint pour la réalisation des buts des deux agents (T non déterminé) ou, même si un plan existe, il ne permet pas d'améliorer le coût de réalisation de deux buts ($T_i > P_i$, $i = 1, 2$). Les figures suivantes permettent d'illustrer la situation :

[17] Dans le modèle de coopération que nous présentons plus tard, l'agent traite de telles situations en utilisant du raisonnement par cas ou des mécanismes de persuasions.

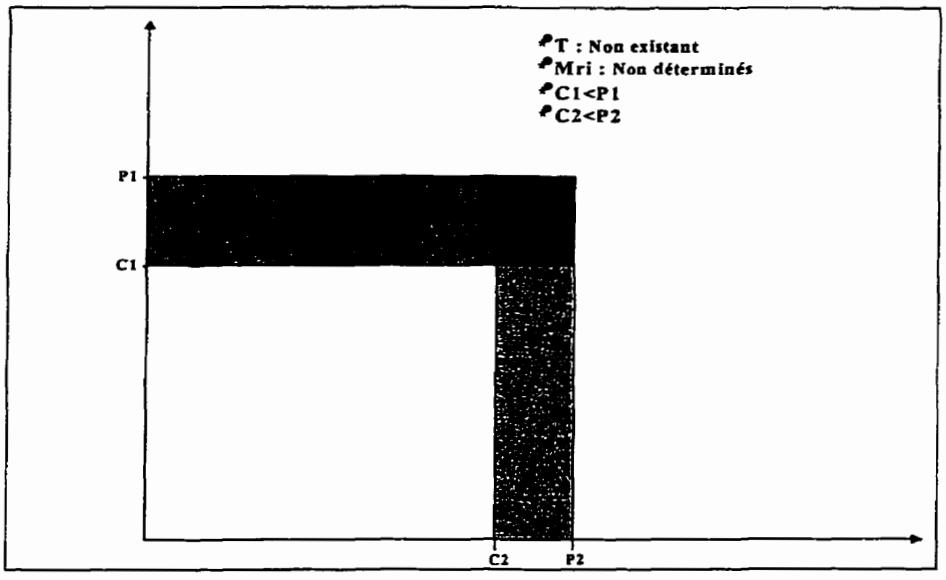


Figure 3.5: Relation d'indépendance, plan conjoint inexistant

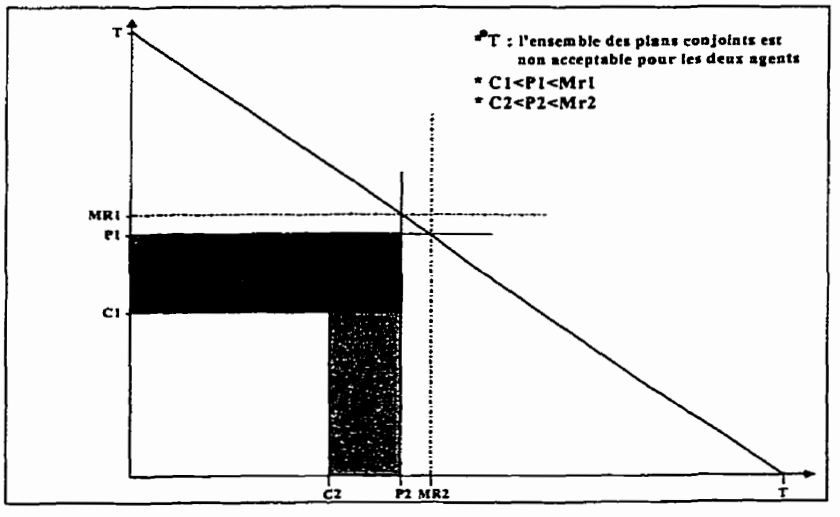


Figure 3.6: Relation d'indépendance, plan conjoint existant mais non profitable

Dans la figure 3.5 : Mr_i *(i = 1, 2)* est indéterminé puisqu'il n'existe pas de plan conjoint. Dans la figure 3.6, même s'il existe un plan conjoint dont le coût est égal à T, on a la contrainte ($T_i > Ci$, $i = 1, 2$), ce qui est non acceptable pour les deux agents.

b- Relation de dépendance mutuelle faible *:* dans une telle situation, chacun des agents peut satisfaire seul son but ($C_i < P_i$, $i = 1, 2$). Cependant, il existe un plan conjoint qui permet aux deux agents de réduire leurs coûts respectifs de réalisation de leurs buts

($T_i < C_i$, $i = 1, 2$). La figure suivante permet d'illustrer la situation de dépendance mutuelle faible :

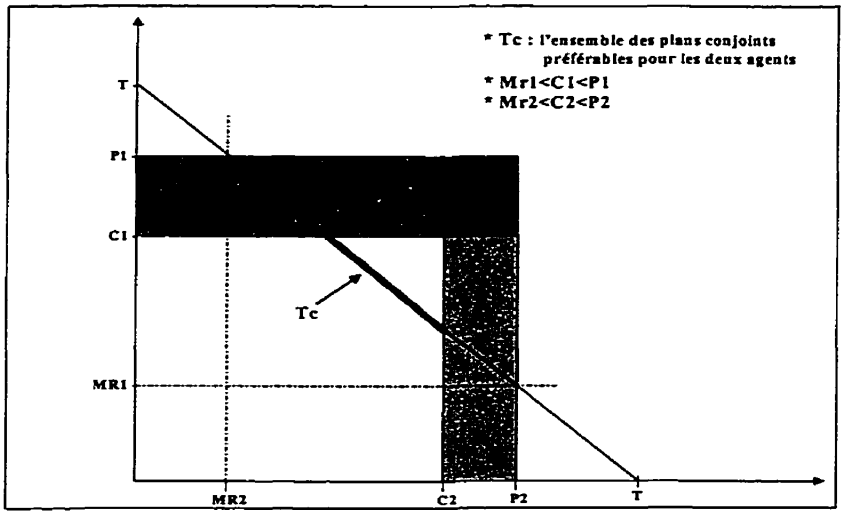


Figure 3.7: Relation de dépendance mutuelle faible

Les deux agents ont intérêt à négocier un plan conjoint, qui en fait correspond à un point du segment *Tc*. La répartition des coûts du plan sur les deux agents correspond aux coordonnées du point correspondant au plan sur le segment *Tc*.

c- Relation de dépendance mutuelle forte : dans une telle situation, aucun des agents ne peut satisfaire seul son but ($C_i > P_i$, $i = 1, 2$). Cependant, un plan conjoint pour satisfaire les buts des deux agents existe ($Tc_i < P_i$, $i = 1, 2$)

La figure suivante permet d'illustrer la situation de dépendance mutuelle forte :

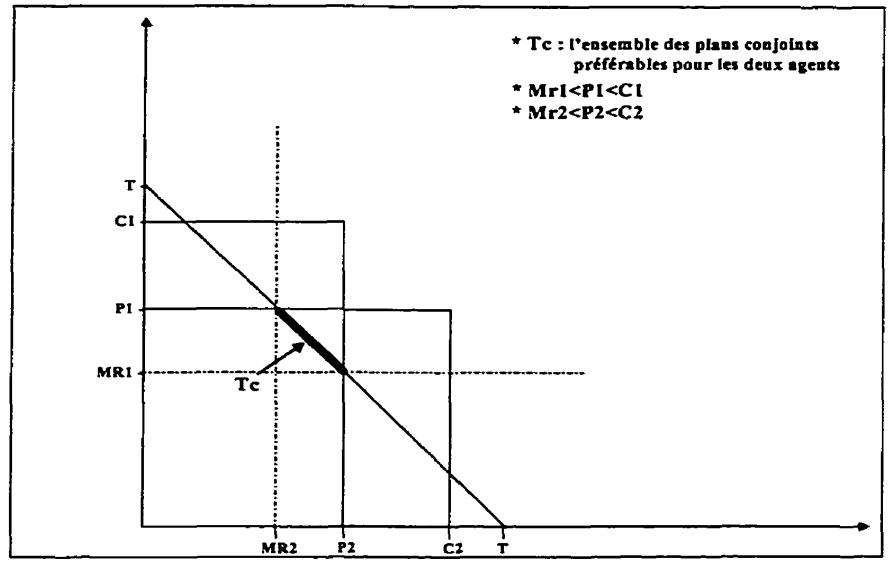


Figure 3.8: Relation de dépendance mutuelle forte (situation coopérative)

Tc est l'intervalle correspondant au sous-ensemble de plans acceptables pour les deux agents.

Un cas particulier d'une relation de dépendance mutuelle forte est celui qui se produit dans une situation de compromis. Aucun des agents ne possède alors un plan individuel pour satisfaire son but. Cependant, il existe un plan conjoint qui permet de satisfaire les deux buts, mais nécessite des deux agents des concessions sur le coût de réalisation de leurs buts. La figure suivante illustre une telle situation :

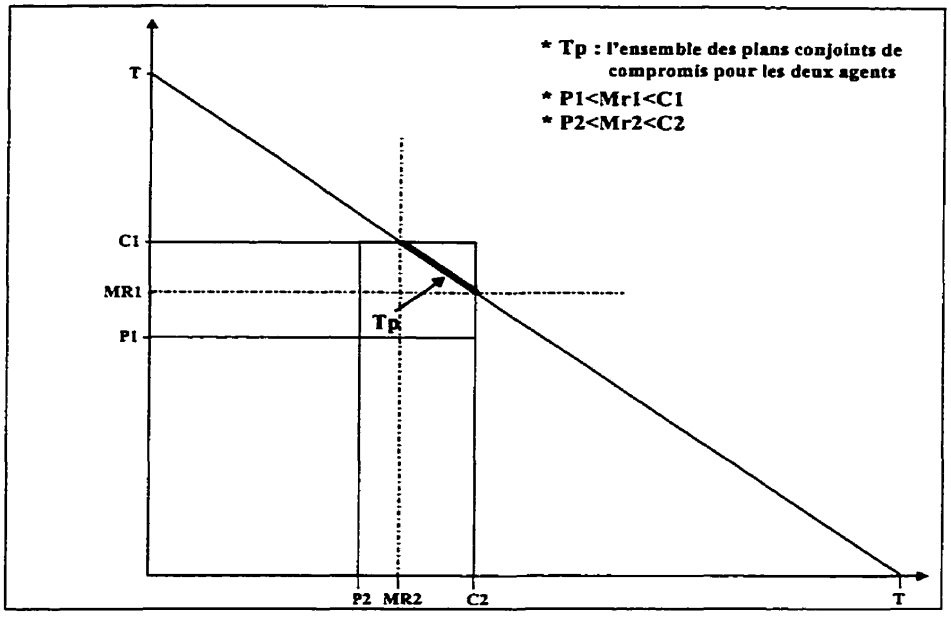


Figure 3.9: Relation de dépendance mutuelle forte (situation de compromis)

d- Relation de dépendance unilatérale : dans une telle situation l'un des agents (i.e., A_1) peut satisfaire seul son but ($C_1 < P_1$), mais l'autre agent ne possède aucun plan pour satisfaire son but ($C_2 > P_2$). Cependant, il existe un plan conjoint qui permet la réalisation des buts des deux agents.

Deux cas peuvent se présenter pour le plan conjoint :

- Il existe un plan conjoint qui, en plus de réaliser les deux buts, permet aussi de réduire le coût de réalisation pour A_1. On est donc en présence d'une situation de négociation coopérative puisque les deux agents ont intérêt à négocier un plan dans l'intervalle *Tc,* figure 3.10,
- Il existe un plan conjoint qui permet de réaliser les deux buts, mais l'agent A_1 doit toutefois accepter une réduction sur le profit à réaliser. Le coût du rôle minimal pour A_1 dans le plan conjoint Mr_1 est supérieur à son coût minimal individuel C_1, figure 3.11.

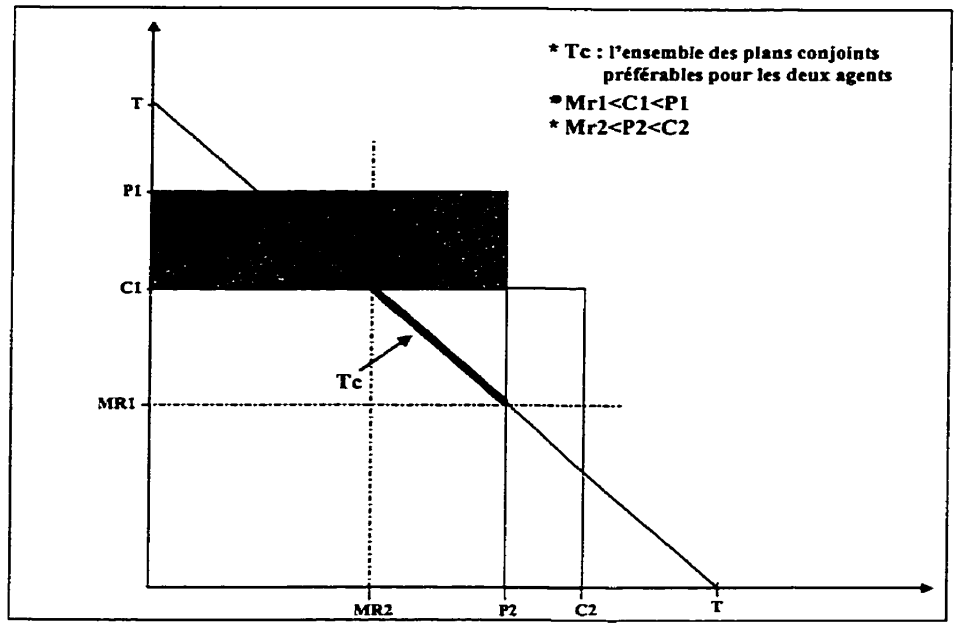


Figure 3.10: Relation de dépendance unilatérale (situation coopérative)

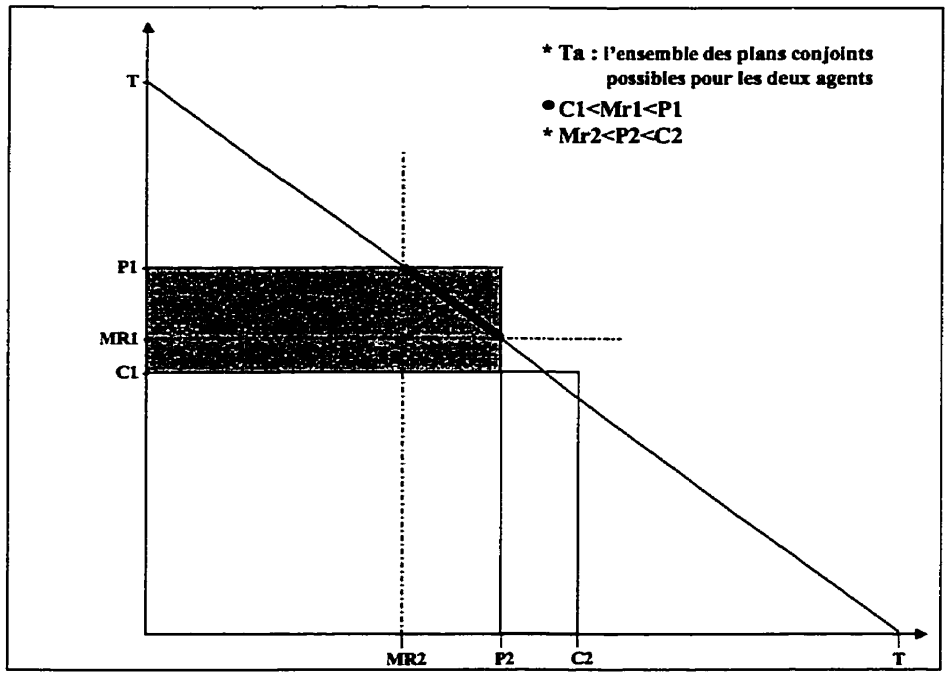


Figure 3.11: Relation de dépendance unilatérale (situation de compromis)

Chapitre 4: Travaux pertinents reliés à la thèse

Comme nous avons cité dans la section 3.1, la problématique traitée dans ce travail est celle du comportement social dans les SMA. Un comportement social peut apparaître dans des sociétés où les membres sont des entités très simples (agents réactifs, cf. 4.5) ou sont des entités complexes (agents cognitifs, cf. 4.5). Deux exigences fondamentales doivent être satisfaites pour assurer un comportement social dans un SMA : (1) une capacité de résolution de conflits entre les différentes composantes du système et (2) une amélioration du profit de l'agent s'il agit de façon sociale, par rapport à celui qu'il obtiendrait en agissant de façon individuelle.

Plusieurs modèles ont été développés pour assurer la coopération des agents dans différents domaines. Diverses techniques sont utilisées dans ces modèles pour spécifier l'organisation et le contrôle des agents dans un environnement SMA. Cependant, trois idées centrales sont considérées dans de tels modèles :

- la difficulté fondamentale d'assurer une coordination globale du comportement des agents à l'aide d'actions basées sur des connaissances locales incomplètes. Le conflit peut surgir à différents niveaux (conflit sur les buts[18] [Sycara 1988a], conflit sur les résultats[19] [Adler 1989], conflit sur les ressources[20] [Sathi 1989], actions qui s'annulent mutuellement, actions redondantes). Le conflit est dû à la nature distribuée de la résolution du problème et aux vues locales et limitées du problème par différents agents. En conséquence, un mécanisme de coopération et d'organisation des agents est incontournable. Pour établir ce mécanisme trois éléments intéressants ont été introduits : (1) le concept de négociation comme mécanisme d'interaction, (2) la

[18] Le conflit de buts surgit quand la réalisation d'un but par l'un des agents empêche un autre d'atteindre le sien, même partiellement. Ce sont les plus fréquemment rencontrés dans la littérature.
[19] Les conflits de résultats surviennent quand plusieurs agents donnent des résultats différents à un tiers pour une même information ou traitement demandé.
[20] Les conflits sur les ressources interviennent lorsque les agents utilisent des ressources communes dont le nombre n'est pas suffisant pour permettre de réaliser leurs buts entièrement ou partiellement.

décomposition du problème en réseau de tâches, (3) un langage commun partagé par tous les agents,

- le besoin d'un protocole de haut niveau [Sproull 1978] pour la coopération entre agents. Dans le même sens que nous considérons des protocoles de communication pour les bas niveaux du système, nous avons besoin d'un protocole pour organiser la résolution de problème dans les SMA. Le but d'un tel protocole est de fournir une organisation qui permet de réduire la quantité d'échanges de messages nécessaires, et de définir les règles de communication ainsi que les messages de communication,
- l'utilité de la négociation comme principe d'organisation. L'élément central dans cette approche est le concept de négociation qui sous-entend une discussion par laquelle les parties intéressées échangent de l'information et arrivent à une entente. Ici, trois aspects sont à noter : (1) il y a échange d'information dans les deux sens et à parts égales, (2) chaque partie évalue les informations selon sa propre perspective et (3) l'entente est conclue par une sélection mutuelle.

Le concept de négociation a été largement utilisé dans les domaines de l'économie, de la gestion, des sciences humaines, de la théorie des jeux et dernièrement dans les systèmes multiagent.

Dans ce chapitre nous allons examiner brièvement les modèles de négociation développés dans les domaines de la théorie des jeux, des sciences humaines, de la théorie de mise aux enchères et, avec un peu plus en détail, les modèles computationnels implantés dans des SMA.

4.1 Théorie des jeux

La théorie des jeux traite de plusieurs aspects dans la coopération des agents : l'élaboration des contrats, la répartition de profit et la résolution de conflits. Plusieurs modèles de négociation ont été proposés dans ce domaine [Rosenschein 1985, Genesereth 1986, Zlotkin 1989, Krauss 1990, Zlotkin 1990, Ephrati 1992, Khedro 1993]. Ces

modèles possèdent certaines propriétés fortement désirées, telles que la capacité de garantir la convergence, l'*équilibre de Nash* (cf. Annexe I) et la *Pareto-optimalité*[21].

Par exemple, Georgeff [Georgeff 83] présente un modèle où les agents sont capables de coopérer sans communication explicite. Le modèle suppose que chaque agent possède une connaissance complète des fonctions d'utilité et des profits de ses pairs. Cette hypothèse est trop restrictive pour appliquer le modèle à des problèmes réels comme le commerce électronique ou la résolution de conflits dans l'interaction d'agents autonomes. Des extensions de ce modèle où les agents peuvent communiquer, où l'information sur les pairs peut être incomplète et où les agents ne sont pas nécessairement honnêtes (peuvent mentir) dans leur communication, ont été proposés par d'autres chercheurs [Gmytrasiewicz 91a, Gmytrasiewicz 91b, Rosenschein 1985, Zlotkin 1990, Zlotkin 1991]. Cependant, ces extensions supposent toujours que les agents ont la capacité d'évaluer la matrice de profits de leurs pairs et de comparer les profits potentiels pour toutes les solutions alternatives. Le temps de calcul dans de tels modèles peut augmenter de façon exponentielle en fonction des solutions à comparer, ce qui les rend inappropriés pour des situations complexes.

Les travaux les plus représentatifs dans le contexte de la théorie des jeux sont ceux effectués par Zlotkin et Rosenschein [Zlotkin 89, Zlotkin 90, Zlotkin 91, Zlotkin 93a, Zlotkin 93b, Zlotkin 93c, Rosenschein 85, Rosenschein 94]. Les auteurs proposent un modèle de négociation formel qui permet aux agents de sélectionner la solution Pareto-optimale qui maximise leurs utilités parmi plusieurs solutions alternatives. Dans ce modèle, les agents communiquent explicitement leurs désirs et peuvent parfois faire des concessions afin d'établir un contrat avec leurs pairs. Un contrat peut concerner aussi bien la répartition de profit que la répartition de tâches à effectuer dans le cadre d'un plan conjoint permettant aux agents de satisfaire leurs buts. Les contrats peuvent être purs ou mixtes.

Un contrat pur est défini de la façon suivante : étant donné une rencontre entre deux agents A_1 et A_2 ayant à exécuter les tâches T_1 et T_2 pour satisfaire leurs buts. Un ***contrat***

[21] Ces notions sont définies peu loin dans cette section.

pur δ relatif à un plan conjoint permettant de satisfaire l'ensemble des buts, est défini par la redistribution des tâches entre les agents en une liste ordonnée (D_1,D_2) telle que $D_1 \cup D_2 = T_1 \cup T_2$ et que chaque agent A_k s'engage à exécuter toutes les tâches dans D_k. Ainsi, le coût du contrat pour A_k est définit par $Cout_k(\delta) = Cout(D_k)$ et l'utilité du contrat pour A_k est définie par $U_k(\delta) = Cout(T_k) - Cout_k(\delta)$. Un ***contrat mixte*** est un contrat pur pour lequel les deux agents ont une probabilité P d'exécuter le contrat et une probabilité $(1-P)$ d'exécuter son complément (c'est à dire les agents s'échangent les rôles). Ainsi, le coût d'un contrat mixte est défini pour A_1 par : $Cout_1(\delta : P) = P*Cout(D_1) + (1-P)*Cout(D_2)$.

Une conséquence importante de l'utilisation des contrats mixtes est l'élargissement de l'espace de coopération sur lequel les agents peuvent conclure des ententes.

Le protocole de négociation est basé sur les concepts de rationalité individuelle, Pareto-optimalité et espace de négociation. En fait, un contrat δ est ***individuellement rationnel*** si pour tout agent A_k, l'utilité que procure δ à A_k est positive $(U_k(\delta) \geq 0)$. Le contrat est aussi dit ***Pareto-optimal*** s'il n'existe aucun autre contrat ω qui le domine - c'est à dire tel que : $(U_1(\delta), U_2(\delta)) > (U_1(\omega), U_2(\omega), \forall \omega \neq \delta)$. L'***espace de négociation*** est défini par l'ensemble des contrats qui sont individuellement rationnels et Pareto-optimaux.

Le protocole de négociation proposé se définit alors comme suit :

1. à chaque étape $t \geq 0$, les deux agents proposent simultanément leurs offres $\delta_1(t)$ et $\delta_2(t)$ qui doivent satisfaire les contraintes : (1) d'appartenance à l'espace de négociation et (2) que pour tout $A_k \in \{A_1,A_2\}$, $\forall t > 0$ on a $U_k(\delta_k(t)) \leq U_k(\delta_k(t-1))$.
2. la négociation s'arrête à une étape t lorsque les agents arrivent à l'une des deux situations:

   * une entente δ tel que pour chacun des agents on a:

     $U_1(\delta_2(t)) \geq U_1(\delta_1(t))$ et $U_2(\delta_1(t)) \geq U_2(\delta_2(t))$.

   * une situation de conflit où il n'est plus possible d'améliorer l'utilité des agents

     $\forall A_k$ on a $U_k(\delta_k(t)) = U_k(\delta_k(t-1))$.

Plusieurs versions de ce modèle ont été proposés par les même auteurs et par d'autres auteurs [Ephrati 1992, Kraus 1995] selon le contexte d'application : domaine orienté

tâche[22], domaine orienté état[23], domaine orienté valeur[24], contrainte de temps ou influence d'un agent superviseur.

4.1.1 Remarques sur les modèles à base de la théorie des jeux

Le point fort de tels modèles c'est qu'ils permettent de faire coopérer des agents qui sont rationnels et agissent de façon à maximiser leur profit probable. Cependant, dans ces modèles, les agents sont supposés toujours avoir une information complète sur la matrice des profits de leurs pairs. Aussi, ces modèles ne prennent pas en considération l'historique de la négociation et chaque décision est prise indépendamment des autres.

4.2 Théorie de vente aux enchères

La théorie de vente aux enchères (VAE) analyse les protocoles et les stratégies utilisées par les agents durant une VAE. Une session de VAE consiste en un vendeur, plusieurs acheteurs potentiels et un protocole de VAE. En général, la VAE est utilisée dans des situations où le vendeur espère recevoir le plus haut prix pour son produit pendant que les acheteurs essaient d'avoir ce produit avec le coût le plus bas possible.
Plusieurs protocoles ont été proposés dans le cadre de cette théorie [Rasmusen 1989] : enchère anglaise, enchère hollandaise, enchère de Vickrey.
Enchère anglaise (enchère publique) : dans l'enchère anglaise, chacun des agents voit tout ce qui se déroule dans le processus d'enchère et chaque acheteur est libre d'augmenter sa proposition. Lorsque aucun des agents ne veut augmenter sa proposition, l'enchère se termine et le plus offrant gagne l'achat au prix de sa dernière offre. Une valeur minimale est généralement spécifiée pour le montant par lequel une offre doit dépasser celle qui la précède pour qu'elle soit retenue comme nouvelle offre. Une *stratégie dominante* dans ce type d'enchère consiste, pour un agent, à augmenter

---

[22] Les domaines orientés tâche : ce sont les domaines où l'activité de l'agent est définie en terme d'ensemble de tâches qu'il doit effectuer.
[23] Les domaines orientés état : ce sont les domaines où chaque agent a pour but de faire déplacer l'état du système d'un état initial vers un des états finals qui satisfont son objectif.
[24] Les domaines orientés valeur : ce sont les domaines où chaque agent associe une valeur d'utilité à chaque état selon l'importance de ce dernier pour ses buts. Les domaines orientés état est un cas particulier des domaines orientés valeur où la valeur d'utilité est binaire : 1 pour les états finals et 0 pour les autres états.

régulièrement son offre de la plus petite valeur acceptable qui lui permette de dépasser l'offre courante jusqu'à atteindre la valeur maximale qu'il peut offrir.

Enchère de Yankee : l'enchère de Yankee est une version améliorée de l'enchère anglaise. Dans ce protocole, le manager de l'enchère fait son annonce d'ouverture de la session de vente aux enchères, dans laquelle il précise, le prix minimal acceptable et la date de fermeture de la session. Ensuite, le manager passe à l'état d'observation alors que les agents concourent de façon transparente (i.e. chacun des agents a accès aux offres faites et peut en conséquence améliorer ses propres offres). À la fin de la session, le manager traite les offres en les triant par ordre d'importance selon les critères : (1) valeur de l'offre, (2) quantité demandée et (3) temps de soumission de l'offre.

Enchère sur enveloppe scellée : dans ce cas chaque acheteur potentiel soumet une offre sans avoir connaissance des autres offres. L'agent ayant l'offre la plus élevée gagne l'achat et paye le montant de son offre. Ainsi, la stratégie de l'agent est fonction de sa valeur privée maximale, de son estimation des valeurs des autres agents et de la valeur de la dernière offre. En général, il n'y a pas de stratégie dominante dans ce type d'enchère, puisque les agents ne disposent pas d'information sur les autres offres.

Enchère hollandaise (enchère décroissante) : au début, le manager de l'enchère initialise l'enchère avec un prix maximal, puis commence à réduire ce prix par unité jusqu'à ce que l'un des agents acheteurs déclare son acceptation du prix atteint. Du point de vue stratégie, l'enchère hollandaise est équivalente à l'enchère sur enveloppe scellée, puisque dans les deux cas aucune information n'est révélée durant le processus d'enchère.

Enchère de Vickrey : chaque acheteur potentiel soumet une offre sans savoir les offres des autres. L'offre la plus value gagne l'achat, mais ne paye que le prix de la seconde offre. Ainsi, la stratégie d'un agent est fonction de sa valeur privée maximale et de sa croyance sur les valeurs des autres agents. La stratégie dominante dans ce contexte consiste, pour un agent, à offrir sa meilleure valeur sans mensonge. En effet, si l'agent propose une offre supérieure à sa valeur acceptable, il risque de gagner l'enchère avec une seconde valeur aussi supérieure à sa valeur maximale qu'il sera obligé de payer et s'il propose une offre inférieure à sa valeur acceptable, il risque de perdre l'offre avec une

valeur qui était à sa portée. Ce protocole offre deux avantages : (1) les agents sont encouragés à utiliser leur performance maximale ce qui augmente la performance globale du système et (2) les agents ne vont pas perdre d'effort computationnel pour faire des contre-spéculations, puisque cela n'aura aucun effet sur la décision finale de l'enchère. L'enchère de Vickrey avait plusieurs utilisations dans les systèmes informatiques : pour l'allocation des ressources dans les systèmes d'exploitation [Waldspurger 1992, Drexler 1988], pour le contrôle d'environnement informatique [Huberman 1995] ou pour l'allocation de ressources dans les réseaux [MacKie-Masson 1994]

4.2.1 Remarques sur les modèles à base d'enchère

Un des problèmes dont soufrent les quatre protocoles d'enchère est leur vulnérabilité aux collusions qui peuvent être menées par des acheteurs. En fait, les acheteurs peuvent coordonner leurs offres de façon à obtenir l'achat avec un prix minimal. Dans cette perspective, les protocoles d'*enchère sur enveloppe scellée* et *enchère hollandaise* sont les meilleurs puisqu'ils découragent ce type de manœuvre. Dans le protocole d'*enchère anglaise*, la collusion est encouragée du fait que tous les agents reçoivent toute l'information sur le processus d'enchère. Aussi durant le processus d'enchère, chacun des agents peut détecter en toute étape si la collusion est respectée par tous les agents ou non, et si la collusion n'est plus valable, l'agent peut toujours utiliser toute sa capacité d'achat sans rien perdre. Dans les autres protocoles (*enchère sur enveloppe scellée, enchère hollandaise et enchère de Vickrey),* les agents ne reçoivent aucune information durant le processus d'enchère. Ainsi, pour pouvoir former une collusion, les agents ont besoin de s'identifier entre eux avant la soumission de leur offre. Autrement, un autre agent non-membre de la collusion peut gagner l'achat en faisant une offre légèrement supérieure à la valeur fixée par la collusion.

Le protocole de Vickrey n'a pas été adopté dans des contextes de négociation pour agents autonomes ou de commerce électronique, puisque aucun contrôle ne permet d'assurer que le vendeur n'augmente la valeur du prix de la seconde offre. Des mécanismes qui permettent de remédier à cette limite, ainsi que d'autres limites sont discutés dans [Sandholm 1996a]

4.3 Modèles des sciences humaines

Les chercheurs des sciences sociales ont analysé le problème de la négociation dans plusieurs domaines tel que les relations internationales et l'organisation de groupes sociaux [Fisher 1981, Pruitt 1981, DeBono 1971] et définissent pour cela un ensemble de stratégies de négociation. L'ensemble des considérations psychologiques utilisées dans ces stratégies ne sont pas directement applicables aux modèles computationnels, mais certaines tactiques décrites dans ces travaux peuvent être adaptées et implantées. Par exemple, Pruitt [Pruitt 1981] propose une stratégie où l'objectif global est décomposé en petites composantes qui seront par la suite traitées selon leur ordre de priorité dans des sessions de négociation séparées. La même tactique a été utilisée par Sathi et Fox [Sathi 1989] qui ont implanté dans leur modèle plusieurs opérateurs de composition en s'inspirant des travaux de Pruitt. Aussi DeBono [DeBono 1971] définit une stratégie de résolution de conflit où il propose un mécanisme pour la génération de solution innovatrice en explorant les différents liens possibles pour dénouer l'impasse. La même idée a été utilisée par Sycara pour définir son modèle de négociation créative [Sycara 1987].

4.3.1 Remarques sur les modèles des sciences humaines

Les modèles de négociation définis dans les sciences humaines prennent en considération des facteurs psychologiques qui ne peuvent être implantés dans des modèles computationnels. Cependant, ils constituent une source intéressante pour la construction de modèles créatifs. En effet, la négociation dans les sciences humaines supporte l'échange d'information imprévisible et cela dans des environnements où les agents sont hétérogènes et où chacun raisonne selon sa perspective locale de la situation.

4.4 Résolution de conflit par vote

Le vote est le mécanisme le plus populaire qui permet aux humains de conclure sur des décisions. Plusieurs méthodes de vote ont été analysées pour des systèmes multiagent. La résolution de conflit y est faite en permettant aux agents de coordonner leurs plans en

décidant sur les actions à prendre par vote. Ainsi, la coordination est explicite puisque tous les agents participent au vote. L'une de ces méthodes est le mécanisme de vote basé sur la *taxe de Clarke* [Ephrati 1991, Ephrati 1996]. Ce mécanisme permet d'élaborer un consensus sur les actions qui doivent être accomplies par les agents sans stratégie de négociation explicite. En fait, l'environnement multiagent est composé d'un groupe d'agents qui interagissent dans un système qui peut passer d'un état à un autre en fonction des actions prises par les agents. Les agents associent une valeur d'utilité à chaque état possible du système et leur but est de satisfaire leurs prédicats (qui consistent à rendre un ensemble d'états vrai). Les agents décident de l'état auquel l'environnement sera amené par vote sur les différents états possibles à chaque étape.

Selon la procédure de *taxe de Clarke*, à chaque étape, le vote d'un agent consiste à annoncer la valeur d'utilité qu'il associe à chacun des états possibles. À la fin du vote et après détermination de l'état gagnant, une taxe (amende) est infligée à chaque agent en fonction de son vote.

La taxe est calculée de la manière suivante : après le vote, l'état qui obtient la valeur maximale (notée par *EG*) est déclaré gagnant, puis pour chaque agent on détermine l'état qui aurait avoir gagné si cet agent n'avait pas voté. Cela est fait en retranchant la valeur du vote de l'agent, puis on détermine l'état gagnant dans cette situation (noté par *EA*).

Si l'état *EA* est le même que l'état *EG*, l'agent reçoit une taxe nulle, sinon l'agent reçoit comme taxe la différence entre la valeur de *EG* et la valeur de *EA*. Pour dépenser la taxe collectée auprès des agents, les auteurs proposent de la distribuer équitablement sur les agents qui n'ont pas participé au vote.

Le processus d'évolution du système d'un état à un autre s'arrête lorsque le processus de vote ne fait pas changer le système d'état.

La procédure de *taxe de Clarke* incite les agents à voter, sans mensonge, avec la valeur réelle qu'ils accordent à chaque état. En effet, si un agent essaie d'augmenter sa valeur pour un état, il prend le risque de recevoir une taxe élevée si l'état ne gagne pas, et s'il essaie de réduire cette valeur, il prend le risque que l'état qui satisfait ses prédicats ne gagne pas.

4.4.1 Remarques sur la résolution de conflit par vote

Le mécanisme de vote à base de la *taxe de Clarke* présente plusieurs limites, parmi lesquelles on peut citer :

- La complexité dans la formation des groupes de votes et la nécessité d'un mécanisme pour maintenir les dettes des agents.
- Le nombre important d'états et d'actions à évaluer peut entraîner une augmentation exponentielle du temps de calcul.
- Un choix arbitraire s'impose pour les actions ayant la même valeur pour un agent.
- Chaque agent est obligé de révéler toutes ses préférences à chaque étape, ce qui est ni nécessaire ni désirable.

4.5 Approche computationnelle pour la négociation

Depuis les premiers travaux en intelligence artificielle distribuée, les auteurs ont analysé et implanté des modèles pour faire coopérer les composantes de leurs systèmes. De telles composantes vont de simples modules de programme [Esmahi 1993] à de complexes systèmes experts [Matwin 1989] ou agents intelligents [Smith 1981]. Pour différentes raisons (conceptuelles, architecturales, et d'applications), les chercheurs dans le domaine d'IAD font la distinction entre agent cognitif et agent réactif [Werner 1992, Carley 1998]. En effet, les agents réactifs sont fondés sur des modèles d'organisations biologiques ou éthologiques, comme les sociétés de fourmis [Drogoul 1992]. Dans ces sociétés, nous pouvons observer l'émergence de quelques comportements « intelligents », rendus évidents au niveau de la société, mais non pas au niveau agent. En fait, un agent réactif fonctionne selon un modèle Stimulus / Réponse, sa représentation de l'environnement et des autres agents n'est pas explicite, il n'est capable ni de se rappeler de ce qu'il lui est arrivé dans le passé ni de planifier ses actions dans le futur. Chaque agent prend connaissance des actions et des comportements des autres en percevant les modifications dans l'environnement. Il n'y a pas de communication directe entre les agents.

Quant aux agents cognitifs, ils sont fondés sur des modèles d'organisations sociales humaines, telles que les groupes [Cavendon 1997], les hiérarchies [Werkman 1990], et

les marchés [Malone 1988]. Les agents ont une représentation explicite de l'environnement et des autres agents. Au contraire des agents réactifs, ils ont une mémoire, c'est à dire, ils peuvent raisonner sur le passé et planifier leurs actions dans le futur. En ce qui concerne leurs interactions, ils communiquent directement par l'envoi de messages. Ainsi, le comportement des agents cognitifs et le choix de leurs actions sont guidés par leurs intentions représentées en terme de buts explicites.

Le choix entre agents réactifs et agents cognitifs peut être expliqué par le compromis efficacité / complexité. Du fait que leur comportement est totalement « câblé », les agents réactifs ont une performance plus efficace en ce qui concerne le temps de réponse du système. Mais aussi leurs limites découlent de ce câblage du comportement, puisque cela entraîne un manque de flexibilité dans la perception et le raisonnement, ce qui les empêche de faire le bon choix au bon moment.

Dans le cadre de ce travail, nous ne nous intéressons pas aux agents réactifs, mais à titre comparatif nous présentons brièvement dans la section suivante un modèle de coopération d'agents réactifs proposé par Ferber [Ferber 1996] pour la simulation d'une colonie de fourmis. Dans les autres sections, nous présentons des modèles de coopération pour agents cognitifs.

4.6 Modèle de coopération pour agents réactifs

Les agents réactifs ont été utilisés dans plusieurs domaines : simulation du comportement social des animaux [Denenbourg 1991], robotique [Brooks 1990] et résolution de problème [Ferber 1996].

Dans la société d'agents réactifs, les différences spatiales sont transformées en structure organisationnelle et différenciations sociales. En effet, la propagation des stimulus et des influences réciproques est fonction de la distance entre les agents. Ainsi, le comportement des agents est déterminé par leur relative position dans la structure topologique de la société.

L'émergence du comportement social et de l'état stable de la société d'agent résulte de la force des mécanismes de réactions face aux stimulus. Les réactions positives encouragent

la création de diversité au sein des agents, alors que les réactions négatives jouent le rôle de régulateur en imposant une force conservatrice sur la structure sociale. Dans le SMA à base d'agents réactifs, les réactions sont de deux types :

- la réaction locale de chaque agent, qui est définie par le concepteur et fait partie de la construction de l'agent,
- la réaction globale de la société, qui est le résultat de l'interaction entre les agents et qui n'est pas définie au niveau agent.

4.6.1 Mécanisme de communication

Les agents communiquent par propagation de signaux dans leur environnement composé de places libres, d'obstacles et d'autres agents. Chaque agent possède son propre stimulus qui joue le rôle de sa signature personnelle. La modification dans l'état d'un agent engendre un signal qui est propagé récursivement sur les places adjacentes, mais dont l'intensité diminue en fonction de la distance parcourue. Ainsi, la diffusion du stimulus dépend des places, chaque place définit une fonction de propagation $f_p(v)$ où v est l'intensité du stimulus à propager. Lorsqu'une place reçoit un stimulus s qu'il faut propager, elle calcule $v'=f_p(strength(s))$, mémorise v' dans son stimulus et demande à sa voisine[25] de propager un nouveau stimulus s' de même nom que s mais d'intensité v'. Une place ne peut mémoriser qu'un seul stimulus, si deux stimulus de sources différentes arrivent à la même place, seul celui ayant l'intensité la plus forte est prise en considération. L'algorithme de propagation des stimulus se présente comme suit :
soit s_n le stimulus à propager, s_s son intensité, *stim(p)* la fonction qui retourne le nom du stimulus contenu dans p, $strength(p,s_n)$ la fonction qui retourne l'intensité de s_n dans s_p, et $add(p, s_n, s_s)$ la fonction qui permet d'ajouter dans p le stimulus s_n avec l'intensité s_s.

1- $propagate(p, s_n, s_s)$ = if $s_n \in stim(p)$ then

(if $f_p(s_s) > strenth(p,s_n)$ and $(f_p(s_s)>0)$ then $add(p, s_n, f_p(s_s))$;

2- for each $p' \in neighbors(p)$ do $propagate(p', s_n, f_p(s_s))$.

---

[25] Le voisinage est défini par la formule : voisinage(place(x,y)) = { p| p = (place(x+a, y+b), -1≤a≤1, -1≤b≤1, p≠place(x,y) }.

4.6.2 Connaissances de l'agent

Les connaissances d'un agent de la colonie sont réduites à l'environnement personnel de ce dernier, constitué de l'ensemble des places desquelles l'agent peut recevoir des stimulus. En fait, l'agent n'a pas besoin de connaître davantage son environnement puisque son comportement est stéréotypé par un processus stimulus / réponse.

4.6.3 Comportement de l'agent

Le comportement d'un agent est défini par un ensemble de tâches qui modélisent les « réflexes » de l'animal. Chaque tâche est reliée à un stimulus particulier et est composée d'un ensemble de primitives construites par le programmeur. À chaque tâche sont associés :

- un nom, qui est généralement celui du stimulus qui la déclenche,
- un poids, qui spécifie l'importance accordée à la tâche au sein de l'agent, ce poids peut être mis à jour par un mécanisme d'apprentissage (processus de renforcement),
- un seuil, au-dessous duquel la tâche ne peut pas être déclenchée par un stimulus,
- un niveau d'activité, calculé quand la tâche devient active,
- deux séquences de primitives, qui sont exécutées respectivement quand la tâche devient active ou inactive.

Pour sélectionner la tâche à activer, l'agent procède selon les étapes suivantes :

1. *détection* : l'agent collecte les stimulus et élimine ceux qui ne correspondent pas à une tâche,
2. *sélection* : l'agent calcule le niveau d'activation de chaque tâche en multipliant l'intensité du stimulus par le poids de sa tâche correspondante. Les tâches dont le niveau d'activation dépasse leurs seuils et dépasse le niveau d'activation de la tâche courante sont sélectionnées,
3. *activation* : l'agent choisit la tâche ayant le plus grand niveau d'activation et procède à son activation.

4.6.4 Processus de renforcement dans le comportement des agents

Le renforcement dans le comportement des agents correspond au fait naturel observé chez plusieurs espèces d'animaux défini par l'assertion : « plus un agent exécute une tâche, plus il se spécialise dans l'exécution de cette tâche ». Ce mécanisme est utilisé dans ce modèle pour introduire une organisation sociale chez les agents de la colonie de fourmis. En effet, le processus de renforcement est déclenché après la défection de chaque tâche pour mettre à jour son poids de la façon suivante : Si la tâche désélectionnée a été arrêtée de façon normale, alors son poids sera augmenté de la quantité totale de l'incrément. Sinon, l'incrément sera relatif à la durée d'activation de la tâche.

4.6.5 Remarques sur le modèle réactif

Cette brève présentation du modèle de la colonie de fourmis, nous permet de percevoir que le comportement social chez les agents réactif émerge d'une part des mécanismes de réaction des agents face aux stimulus, ce qui en fait conduit à une sorte de compétition entre les agents pour déclencher les tâches, et d'autres part de la stabilité de distribution du travail au sein de la colonie entraînée par les mécanismes de renforcement.
Notre travail de recherche consiste en l'exploration de mécanismes de coopération basée sur l'utilisation de communication intentionnelle et de raisonnement sur l'état mental des agents pour induire un comportement social chez les agents. Ce qui nous a conduit à ne pas examiner en détail les modèles de coopération réactifs.

4.7 Protocole Contract-Net

4.7.1 Description du protocole Contract-Net

Dans le protocole *Contract-Net (Contract-Net Protocol — CNP)* [Smith 1981, Smith 1988a, Smith 1988b], un contrat est une entente explicite entre un agent qui annonce une tâche (le manager) et un agent qui est prêt à exécuter la tâche (le contractant). Le manager est responsable de la gestion de la tâche et du traitement des résultats, alors que le contractant est responsable de l'exécution de la tâche et de la transmission des résultats au manager.

La négociation est initialisée quand un agent décide de confier, à un autre agent, l'exécution d'une tâche pour laquelle il ne dispose pas des capacités ou connaissances nécessaires pour bien l'accomplir. Quand cela arrive, l'agent annonce l'existence de la tâche par la diffusion d'un message *'Task-Anouncement'* et devient donc le manager pour cette tâche.

Les agents libres (contractants potentiels) évaluent leur niveau d'intérêt pour les tâches qu'ils reçoivent en utilisant des procédures d'évaluation appropriées au domaine. Si une tâche est jugée suffisamment intéressante, l'agent soumet une offre par un message *'Bid-Message'*.

Un manager peut recevoir plusieurs offres qu'il évalue selon une procédure spécifique à la tâche et sélectionne la meilleure. La sélection est communiquée à l'agent concerné par un message *'Award-Message'*. L'agent sélectionné assure donc la responsabilité d'exécution de la tâche et devient le contractant pour cette tâche. Éventuellement le contractant peut partitionner la tâche en sous-tâches et les sous-contracter avec d'autres agents, ce qui conduit à une hiérarchie de contrôle.

Un message *'Report'* est utilisé par le contractant pour informer le manager que la tâche a été exécutée partiellement (*Interim-Report*) ou complètement (*Final-Report*).

Le manager peut mettre fin à un contrat en envoyant un message *'Termination'* au contractant qui, à son tour, arrête l'exécution du contrat et termine tous les sous contrats.

Le diagramme d'états suivant résume le processus d'établissement de contrat :

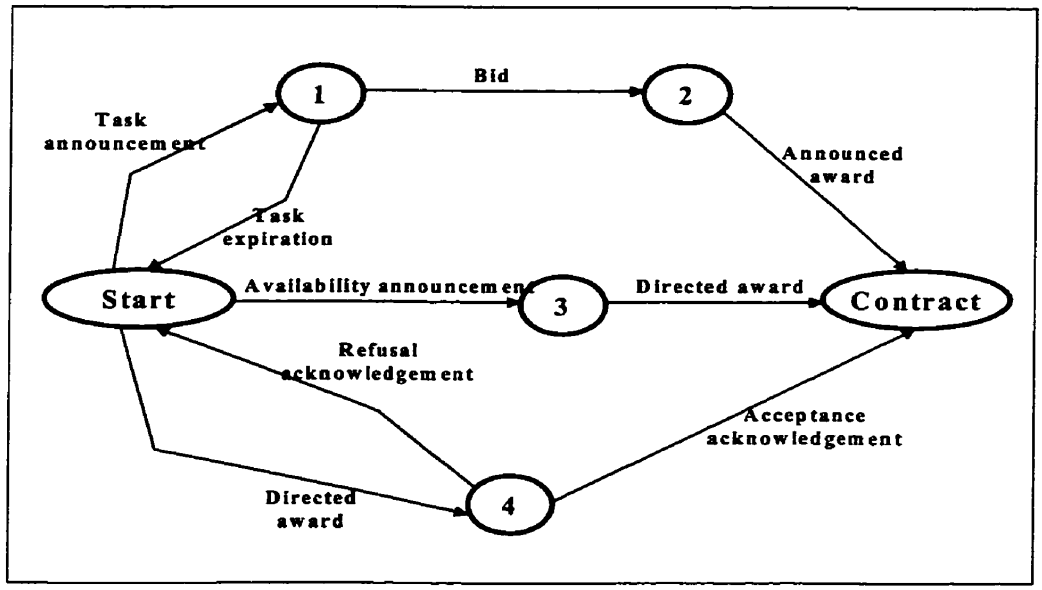


Figure 4.1: Diagramme d'états du Contract-Net

4.7.2 Limites du Contract-Net

Nous pouvons noter parmi les limites du Contract-Net tel que présenté ci dessus, l'absence des notions de coût et d'utilité chez l'agent. Cela suppose que le protocole, tel que présenté, ne fonctionne que dans des environnements où les hypothèses de bénévolat, de performance et de modularité sont vérifiées. Cela suppose aussi qu'il existe un *but commun* partagé par les agents et que des critères de succès sont spécifiés. Beaucoup de domaines ne satisfont pas ces conditions (commerce électronique, ingénierie concurrente).

Puisque l'agent ne peut avoir qu'un seul contrat à un moment donné et doit attendre de terminer la tâche en cours avant de soumettre une autre offre, cela entraîne deux problèmes. D'une part, il y a le coût de maintenir une longue liste d'offres reçues et non traitées et d'autre part, il y a le coût généré par l'attente d'une confirmation de contrat alors qu'il est libre. Il aurait pu avoir déjà planifié un contrat sans cette contrainte.

On observe aussi l'absence de symétrie dans la négociation étant donnée l'absence de possibilité de faire des *contre-propositions* (négociation de *ce qui doit être fait* et non pas

seulement *qui le fait*). Cela limite le champ d'application du protocole à des domaines où on peut définir des hiérarchies de tâches.

L'instanciation de la plate-forme est laissée au complet au concepteur puisque le protocole ne fournit que les attributs pour définir une telle instanciation (i.e. '*Eligibility-Specification*' dans les messages '*Announce*' et '*Bid*'). C'est au concepteur de définir le contenu en fonction du domaine spécifique.

Le modèle de coopération, spécifié par le protocole, est centré sur le partage de tâches mais dans certains domaines (Design collaboratif, planification distribuée, reconnaissance de la parole ou d'image) la coopération est plutôt basée sur le partage de résultats. Un modèle qui supporterait aussi le partage de résultats serait intéressant.

4.7.3 Contract-Net à base des coûts marginaux

La version du CNP, présentée dans le paragraphe précédent, suppose que les agents sont totalement coopératifs. En conséquence, aucun modèle formel n'a été discuté pour faire les annonces, les offres et les attributions de tâches. La présente extension du CNP [Sandholm 1993, Sandholm 1995, Sandholm 1996b] est basée sur un modèle où les agents calculent localement leurs coûts marginaux pour la réalisation des tâches. Le choix du contractant est basé seulement sur ces coûts. Un tel mécanisme permet d'étendre le CNP aussi bien aux agents coopératifs qu'aux agents compétitifs.

a- Annonce de tâche : dans l'annonce d'un ensemble de tâches, le manager précise un paramètre $C^{anounce}$ qui est en fait le paiement maximum qui peut être exigé par un agent qui soumet une offre. Ce qui permet d'éviter les soumissions non profitables et le gaspillage des ressources computationnelles et de communication. Pour un agent qui veut faire une annonce, on a : $C^{anounce} = C_i^{remove}(T^{anounce}|T_i)$

où $C_i^{remove}(T^{anounce}|T_i)$ est le coût marginal, pour l'agent A_i, pour retirer l'ensemble des tâches $T^{anounce}$ de sa solution locale qui comprend toutes ses tâches T_i.

Si on dénote par $C_i(T)$ le coût optimal pour la réalisation de T par l'agent A_i, on a :

$C_i^{remove}(T^{anounce}|T_i) = C_i(T_i) - C_i(T_i \backslash T^{anounce})$ où «\»: est la différence ensembliste.

b- Soumission d'offre : pour soumettre une offre, l'agent consulte le prix mentionné dans l'annonce, et si le prix $C^{anounce}$ mentionné est supérieur au prix que va lui coûter la

réalisation de cette tâche (C^{bid}), il envoie une offre avec son coût C^{bid}; autrement aucune offre n'est envoyée. Le prix d'offre pour un agent A_j est: $C^{bid} = C_j^{add}(T^{anounce}|T_j)$

où $C_j^{add}(T^{anounce}|T_j)$ est le coût marginal généré par l'ajout de l'ensemble des tâches $T^{anounce}$ à la solution locale de A_j qui comprend toutes les tâches T_j.

$C_j^{add}(T^{anounce}|T_j) = C_j(T_j \cup T^{anounce}) - C_j(T_j)$

où $C_j(T)$ est le coût optimal pour la réalisation de la tâche T par l'agent A_j.

c- Attribution de tâches : avant d'attribuer un ensemble de tâches, l'annonceur attend l'écoulement du temps spécifié dans sa demande. Ainsi, plusieurs contractants potentiels auront du temps pour soumettre leurs offres. Juste avant de commencer l'évaluation des offres reçues, l'agent vérifie pour une dernière fois si le contrat demeure encore profitable pour lui, et révise son prix d'attribution des tâches annoncées en fonction du nouvel ensemble de tâches courant T_i' qui est probablement différent de T_i qui était au moment de l'annonce.

Maintenant son nouveau prix d'attribution est: $C^{award} = C_i^{remove}(T^{anounce}|T_i')$

où $C_i^{remove}(T^{anounve}|T_i')$ est le coût marginal pour l'agent A_i de retirer $T^{anounce}$ de sa solution locale comprenant toutes ses tâches T_i'.

$C_i^{remove}(T^{anounce}|T_i') = C_i(T_i') - C_i(T_i'|T^{anounce})$

où $C_i(T)$ est le coût de la solution optimale pour la réalisation de T par l'agent A_i.

Ainsi, si C^{award} est supérieur au plus petit coût offert, l'agent envoie un message d'attribution à celui qui a offert ce plus petit coût.

Par convention, les auteurs [Sandholm 1996b] utilisent $(C^{bid}+C^{award})/2$ comme prix réel du contrat que le manager paie au contractant.

4.7.4 Remarques sur le Contract-Net

Le protocole Contract-Net est l'un des modèles de coopération dans les SMA des plus utilisés et des plus complets. La notion d'utilité, introduite dans l'extension, vient remédier au problème de motivation chez les agents compétitifs. Le système respecte aussi la notion d'autonomie qui est une des caractéristiques fondamentales des agents. Mais le protocole reste toujours incomplet et on peut noter les limites suivantes :

- l'absence de raisonnement sur les états mentaux des pairs pendant la coopération et l'absence de mécanismes de persuasion,
- l'exploitation des connaissances sur l'environnement et la dynamique des agents dans la société ne sont pas prises en considération,
- il est aussi facile de montrer que le processus de négociation, décrit par le protocole, ne garantit pas une affectation globalement optimale entre les tâches et les agents. En effet, cela est dû d'une part au fait qu'un contractant une fois engagé dans une tâche ne fait pas l'évaluation d'autres offres qui risquent d'être plus intéressantes et, d'autre part, aucun mécanisme de désengagement des contrats qui ne sont plus profitables n'est discuté. Le problème de l'optimalité de la solution peut être illustré par l'exemple suivant : considérons deux managers, A et B, ayant annoncé deux tâches et deux contractants potentiels, X et Y, ayant soumis des offres à ces tâches avec des coûts appartenant à l'intervalle [0,10].

	A	B
X	9	8
Y	8	2

La solution choisie est d'affecter Y à A et X à B qui n'est pas la solution optimale (coût=16), alors que la solution optimale est d'affecter X à A et Y à B (coût=11). En fait ce problème est général à tout système distribué puisque les décisions sont locales. Cependant, on peut imaginer des hypothèses pour réduire l'effet du problème. Par exemple, permettre des échanges entre A et B ou mieux encore, si le système supporte les contrats de groupes et les agents A et B soumettent un contrat de groupe,

- finalement, on peut aussi noter l'absence d'attitude de risque chez les agents, du fait qu'on suppose que les agents sont totalement coopératifs et qu'ils honorent parfaitement leurs contrats. Ce qui n'est pas le cas dans les domaines où l'incertain est présent et où les agents sont compétitifs.

4.8 Négociation à base des connaissances

La négociation à base des connaissances (*Knowledge Based Negotiation — KBN*) a été proposée par Werkman [Werkman 1990, Werkman 1993] dans un contexte de design collaboratif. Elle est basée sur l'utilisation des aspects spécifiques des connaissances sur le domaine pour résoudre les conflits de façon progressive. Dans cette technique, les conflits sont résolus en utilisant des représentations de connaissances partagées par les agents en conflit (*Sharable Agent Perspectives*)[26]. Chaque agent maintient un schéma indexé des domaines du SMA. Un tel schéma permet à un agent en conflit de se référer à la perspective de son négociateur lors de la génération d'une contre-proposition. Ainsi, chaque agent peut négocier de façon raisonnable sans avoir besoin des connaissances complètes des autres agents.

Les conflits sont résolus par un agent superviseur en utilisant ses connaissances de médiation et d'arbitrage et les connaissances globales partagées. Dans un premier temps le superviseur essaie de faire une médiation entre les agents afin de les aider à sortir de la situation d'impasse. Si les agents n'arrivent pas à un accord sur une solution, le superviseur fait son arbitrage et force le comportement des agents de sorte à générer une solution valable.

4.8.1 Remarques sur la KBN

L'utilisation d'un raisonnement sur les connaissances des agents (même si très limitées) et sur les connaissances de sens commun partagées par tous les agents, permet l'introduction d'une composante importante, qui est en réalité partie intégrante du processus de négociation lui-même. Cependant, pour la KBN on peut noter les limites suivantes :

- l'architecture centralisée du tableau noir limite l'autonomie des agents,
- l'absence de la notion d'utilité et de motivation chez les agents (ils sont supposés parfaitement coopératifs),

[26] Les SAP (Sharable Agent Perspectives) correspondent aux connaissances communes accessibles par tous les agents et ne comprennent aucun détail qui relève de l'expertise de chaque agent.

- l'agent ne possède pas de composante responsable du raisonnement de coopération. Mais la coopération découle de façon implicite du processus de contrôle de l'agent lui-même,
- les règles de médiation ou d'arbitrage sont définies à l'avance avec la construction du système et ne dépendent pas du contexte.

4.9 Négociation dirigée par les contraintes

Le modèle de négociation à base de contraintes (*Constraint-Directed Negotiation — CDN*) [Sathi 1989] présente le processus de négociation comme une technique de résolution à base de contraintes pour les problèmes de ré-allocation de ressources. En effet, chaque agent possède un certain nombre de ressources et doit réaliser un ensemble de tâches (éventuellement parallèles). C'est en fonction de sa charge de travail[27] qu'il détermine s'il a besoin d'acheter des ressources ou d'en vendre d'autres dont il n'a pas besoin.

La négociation est donc modélisée comme une série de contraintes sur les ressources et les tâches. Les solutions sont générées par relaxation des contraintes en fonction de leur importance, leur degré de liberté et leur seuil.

La négociation est organisée en deux phases :

- la phase communication pendant laquelle les agents s'échangent les offres et les demandes de ressources et,
- la phase de marchandage où les ententes sont faites entre agents ou groupes d'agents en utilisant des techniques de composition[28], de reconfiguration et de relaxation.

La construction de solution à base de relaxation et reconfiguration ou la construction de cascades[29] à base de composition est réalisée par le médiateur qui la propose par la suite aux agents.

---

[27] La charge de travail est définie par l'ensemble des tâches que l'agent est engagé à réaliser, soit pour la réalisation de ses propres buts ou pour aider d'autres à la réalisation de leurs buts.

[28] Dans le contexte de ré-allocation de ressources, la composition est le regroupement ou la combinaison d'offres d'achat et de vente, afin de satisfaire des contraintes conditionnelles.

[29] une cascade est un ensemble de deux ou plusieurs transactions qui ne peuvent être exécutées que si elles sont rassemblées.

L'algorithme général du processus de construction des solutions par le médiateur se fait selon les étapes suivantes :

1. pour chaque paire d'offres (achat, vente), le médiateur crée une transaction dans la mémoire commune.
2. chaque agent évalue les transactions associées à ses demandes d'achat en fonction de ses contraintes.
3. chaque agent supprime toutes les transactions qui ne satisfont pas ses contraintes d'exigence fixes.
4. le médiateur utilise ensuite la composition dans l'ensemble des transactions acceptables afin de créer des cascades.
5. s'il existe une cascade acceptable qui ne nécessite pas de reconfiguration et n'est pas en concurrence avec une autre cascade (i.e., ne possède pas une vente ou achat en concurrence avec aucune autre cascade), alors elle est installée. S'il n'y a plus de transaction ou cascade, alors l'algorithme s'arrête.

6. s'il existe un ensemble de cascades acceptables qui ne nécessitent pas de reconfiguration et sont en concurrence entre elles sur des achats ou des ventes, alors le médiateur choisit une cascade ayant le nombre maximum d'annonces. La relaxation est appliquée sur cette dernière pour casser les nœuds. Puis chaque agent installe la cascade choisie et supprime toutes ses concurrentes. Les étapes 5 et 6 sont répétées pour toute cascade acceptable.
7. s'il existe un ensemble de cascades acceptables qui nécessitent la reconfiguration, alors le médiateur choisit celle qui demande moins de reconfiguration et applique la reconfiguration pour en dériver une cascade acceptable. Puis chaque agent installe la cascade choisie et supprime toutes ses concurrentes. Les étapes 5 à 7 sont répétées pour chaque cascade nécessitant la reconfiguration.

4.9.1 Remarques sur la CDN

Même si la notion de contrainte est simple à implanter et à gérer par les agents, la négociation CDN présente plusieurs limites dont on peut citer :

- la grande partie du traitement social incombe au médiateur puisque les agents ont une vision très limitée dans la sélection des transactions et la construction de compositions,
- la négociation est centralisée sur un médiateur,
- il n'y a pas de raisonnement sur l'état mental des agents ni moyen de motivation pour ceux-ci,
- on peut facilement se trouver avec une explosion du nombre de compositions et de reconfigurations à faire, et ce de façon exponentielle avec l'augmentation du nombre d'agents impliqués et de contraintes,
- un agent peut avoir de la difficulté à combiner ses contraintes, ce qui peut créer une situation d'impasse,
- l'agent ne possède aucun mécanisme de préférence sur les offres qu'il reçoit de ses pairs.

4.10 Négociation multi-étapes

La coopération dans le modèle de négociation multi-étapes (*MultiStage Negotiation — MSN*) [Conry 1991] est faite à base de planification distribuée. En effet, les agents s'échangent des sous-plans. Ainsi, ils acquièrent suffisamment d'information sur les plans des autres et modifient leurs actions en fonction de l'impact de leurs décisions locales sur le plan global. Le processus de la MSN peut être résumé comme suit :

1. au début chaque agent identifie sa capacité à résoudre un ensemble de sous-plans et les distribue aux autres agents,
2. chaque agent construit ensuite son plan et le diffuse à tous les agents,
3. puis chaque agent attend de recevoir soit un engagement des autres agents à accepter son plan, soit une réponse spécifiant le problème et l'influence de son plan sur les plans des autres,

4. si un agent reçoit un feed-back négatif des autres, il prend en considération cette information et révise son plan. Ensuite, un nouveau plan est proposé, et ce processus continue jusqu'à arriver à un ensemble de choix consistant.

Durant la négociation, les agents maintiennent deux types de buts :

- les buts primaires, que l'agent doit satisfaire pour la résolution du problème,
- les buts secondaires, générés à base d'un contrat avec d'autres agents. Chaque but secondaire est une solution possible d'un but primaire d'un autre agent.

4.10.1 Remarques sur la MSN

Ce protocole est intéressant dans le sens qu'il permet de trouver une solution au conflit s'il existe un plan qui satisfait tous les buts. Il permet aussi de détecter les cas de sur-contraintes. Cependant, en plus des problèmes déjà mentionnés dans les modèles précédents, comme l'hypothèse de bienveillance des agents, l'absence de raisonnement sur l'état mental des agents, la MSN souffre de deux grands problèmes :

- la surcharge de la communication à cause du nombre exponentiel de sous-plans à échanger et
- l'absence de mécanisme qui garantit la convergence vers une solution globale, surtout avec des agents non coopératifs.

4.11 Discussion

Les approches de négociation présentées dans les paragraphes précédents traitent le problème de coopération dans les SMA selon des vues limitées et généralement complémentaires. Des remarques ont été dressées pour chacun des approches.

De façon générale, le conflit est résolu soit par une évaluation locale à base des fonctions d'utilité (cas du Contract-Net et de la MSN), soit par l'intervention d'un superviseur central (cas des autres approches : KBN et CDN).

Dans ces approches, on peut aussi identifier deux groupes ou, en d'autres termes, deux modèles de coopération: (a) le modèle fondé sur la théorie d'utilité pour assurer la

convergence de la négociation, et (b) le modèle fondé sur les connaissances ou sur des structures organisationnelles prédéfinies.

Pour le modèle fondé sur les connaissances ou sur des structures organisationnelles prédéfinies (i.e., présence de superviseur...), il suppose que l'interaction sociale sert à coordonner un ensemble d'agents, qui ont déjà un but commun à accomplir. L'allocation de tâches entre des agents est préétablie par le concepteur et les différents rôles de résolution sont définis à priori.

Nous croyons qu'une approche fondée exclusivement sur les relations organisationnelles pour modéliser l'interaction sociale a les inconvénients suivants :

- le biais de sur-coopération : l'approche de sociabilité est "top-down". On suppose l'existence d'un superviseur, d'un but social, et d'un problème collectif. En conséquence, les problèmes d'allocation de tâches et de ressources, et plus généralement ceux de coordination des actions sont ignorées par les agents,
- la médiation est un processus très coûteux puisque toute l'information nécessaire doit être explicitement collectée auprès des agents ou calculée par le superviseur. Le superviseur doit aussi connaître suffisamment tous les aspects du domaine pour pouvoir prendre des décisions correctes,
- l'organisation prédéfini du système entraîne l'absence d'une perspective dynamique. Les agents sont tellement au courant des conditions de leurs interactions que le processus d'émergence d'une dynamique (propose / contre-propose) à partir des actions sociales est complètement ignoré,
- l'analyse des relations sociales est faite en supposant que des buts et des croyances communes existent déjà et sont représentés dans l'état mental des agents,
- les structures organisationnelles rigides ne permettent pas de prendre en compte l'évolution des interactions au sein d'une société ouverte.

Pour le modèle fondé sur l'utilité, il utilise comme éléments de base deux théories modernes d'économie : la théorie de l'utilité [Ponsard 1977] et la théorie de la décision [Luce 1967]. La théorie de l'utilité propose un modèle pour caractériser les préférences d'un agent entre les divers états possibles du monde, en reflétant ses désirs de haut

niveau. La théorie de la décision étend cette dernière en permettant de représenter des opérations quantitatives sur des préférences et en rendant possible le calcul des préférences lorsque de l'incertitude est présente (utilité espérée). La théorie des jeux est construite à partir de ces deux théories, et s'intéresse à une source d'incertitude très particulière, celle qui prend en compte l'action d'autres agents.

Nous croyons qu'une approche fondée exclusivement sur l'utilité pour modéliser l'interaction sociale a les limites suivantes :

- les agents sont considérés comme homogènes et autosuffisants, et les interactions sociales ne sont là que pour maximiser l'utilité de chacun ou pour minimiser l'interférence sociale. Ce qui n'est pas le cas dans les systèmes où les agents ont des expertises ou ressources complémentaires,
- les agents sont hypercognitifs : les agents sont considérés savoir tout concernant leurs interactions sociales et leur société,
- manque de limite sur l'autonomie des agents : les agents sont considérés libres dans leur négociation [Muller 1996], les relations de dépendances entre les agents sont ignorées. De même, les normes et habitudes préexistantes ne sont pas imposées pour le système et leurs effets ne sont pas considérés,
- la notion de but n'est pas représentée explicitement, et l'action sociale est purement stratégique. En effet, un agent ne prend en compte que les effets possibles des actions des autres pour prendre ses décisions. Les agents sont considérés comme des entités passives, incapables de changer et d'influencer les actions possibles des autres. C'est justement à partir des actions sociales d'influence et d'adoption qu'on peut expliquer pourquoi une coopération a eu lieu,
- une vision monistique de la communication : la seule fonction accordée à la communication est d'administrer les relations avec les autres pour accomplir ses buts. En fait, on a ignoré la fonction principale de la communication qui est l'influence des autres pour changer un comportement ou pour adopter un but,
- la structure d'une négociation à base de la théorie d'utilité est représentée par une matrice de profits associés aux différentes stratégies. Chaque stratégie représente une

séquence complète de choix pour une séquence particulière de contingences. Si plusieurs choix sont disponibles à chaque étape, on se trouve très vite avec un nombre énorme de stratégies. La situation devient donc intractable (NP-Complète),

- la théorie des jeux suppose que chaque agent connaît la matrice complète de profit de lui-même et de ces pairs, ce qui n'est pas réaliste,
- la formulation de la négociation exclusivement à base d'utilité ne peut pas accommoder les tactiques d'argumentation et de l'incertain qui sont partie intégrante de la négociation.

En conséquence, nous croyons qu'il est possible de fournir une représentation plus significative du processus de coopération, si on exploite la complémentarité entre la notion d'utilité et le raisonnement sur les états mentaux des agents, et si on introduit une représentation explicite de l'activité de négociation. La représentation explicite de la négociation sera introduite par des mécanismes permettant d'exprimer le fait que l'agent A_1 essaie de convaincre l'agent A_2 de la proposition X, ou que les agents A_1 et A_2 sont engagés dans un conflit qu'ils essaient de résoudre avec la médiation de l'agent Z, sans que l'agent Z soit un contrôleur central mais plutôt un agent choisi par les deux agents en conflit de façon dynamique et en fonction du contexte en cours.

Le chapitre suivant présente nos idées sur un tel modèle.

Chapitre 5: Une approche persuasive pour la coopération

Dans l'étude présentée dans les sections précédentes sur la coopération dans les systèmes multiagent, nous avons identifié que la plupart des systèmes sont soit basés sur une organisation prédéfini ou centralisée sur un agent qui s'occupe de la supervision pour aider les agents à résoudre les conflits, soit basés sur des fonctions d'utilité et un protocole qui utilise un mécanisme d'annonce / réponse. La première solution présente le désavantage de centraliser le processus de négociation et limite le champ d'application aux domaines où les agents possèdent un but commun de haut niveau. La seconde limite la façon de gérer les conflits puisque les contrats sont seulement générés à base de l'évaluation des offres et il n'y a aucun mécanisme pour convaincre le partenaire.

Dans ce chapitre nous décrivons un modèle générique de coopération qui utilise l'argumentation pour convaincre les partenaires à changer leurs croyances et leur comportement envers une proposition. Ainsi, ce modèle offre l'avantage de donner plus de possibilité d'avoir des ententes puisque le processus de coopération (cf. 2.5) a été étendu par une phase de persuasion, figure 5.1. Ainsi, après la reconnaissance du potentiel de coopération, les agents essaient d'abord de former une coalition de façon coopérative, mais si le processus de négociation coopérative n'aboutit pas, une phase de négociation persuasive a été ajoutée pour étendre l'espace d'entente entre les agents. Pendant cette phase, les négociateurs utilisent des argumentations pour se convaincre les uns les autres soit directement ou avec l'aide d'un médiateur choisi en fonction du contexte en cours.

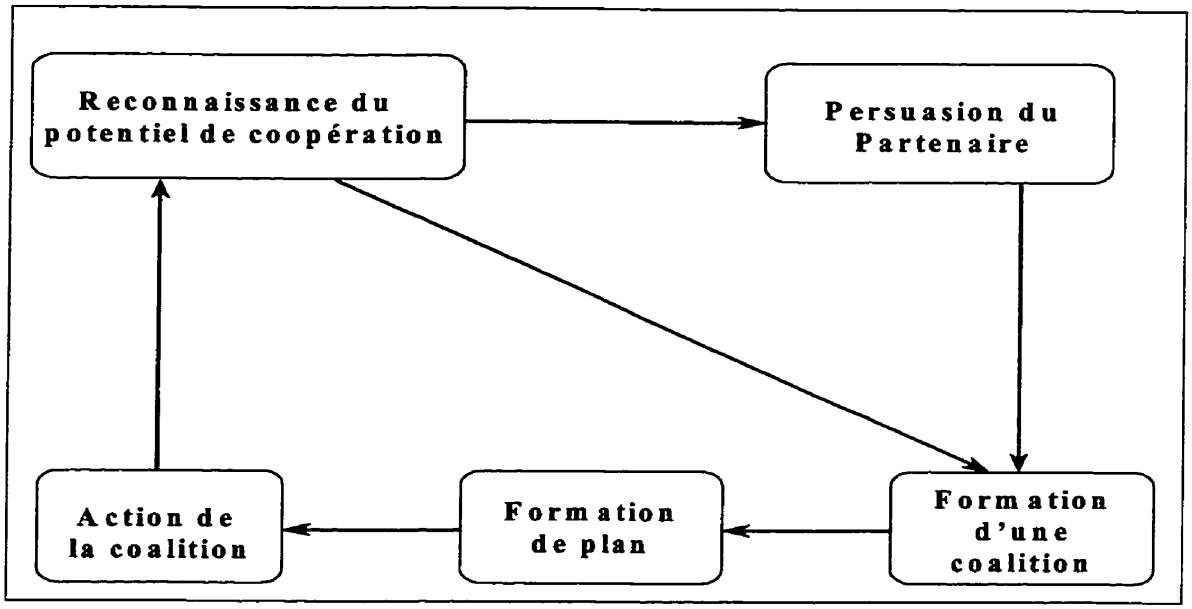


Figure 5.1: Phases de coopération étendues.

Cette extension du processus de coopération nous a amené à considérer d'autres composantes dont dépend le processus de négociation parmi lesquelles on peut citer : (1) les croyances et les attitudes des négociateurs, (2) l'historique de la négociation, (3) le contexte de négociation, et (4) les connaissances et raisonnement nécessaires pour la médiation. Ces composantes ont été intégrées à notre modèle et sont représentées soit par des règles ou par des structures orientées objet où chaque concept est représenté en termes d'ensemble d'attributs dont les valeurs peuvent être numériques, symboliques ou tout autre concept.

Par rapport à l'état mental des agents, la considération de ces nouvelles composantes nous conduit à introduire de nouvelles exigences sur les capacités des agents :

- capacité de raisonner sur les autres,
- capacité d'évaluer si ces buts sont réalisables et si ces plans sont exécutables,
- capacité d'évaluer le profit relatif à une coalition,
- capacité de générer de l'argumentation pour convaincre les autres.

Par rapport à ce qui est proposé par la FIPA [FIPA 1997a] pour l'élaboration de contrat, nous avons ajouté une boucle de persuasion et étendu certaines performatives [Esmahi 1997, Bernard 1998] pour supporter les différents types de contrats (cf. 5.3.2). En

utilisant la représentation par diagramme de transition d'états, la procédure d'élaboration d'un contrat se présente alors comme suit :

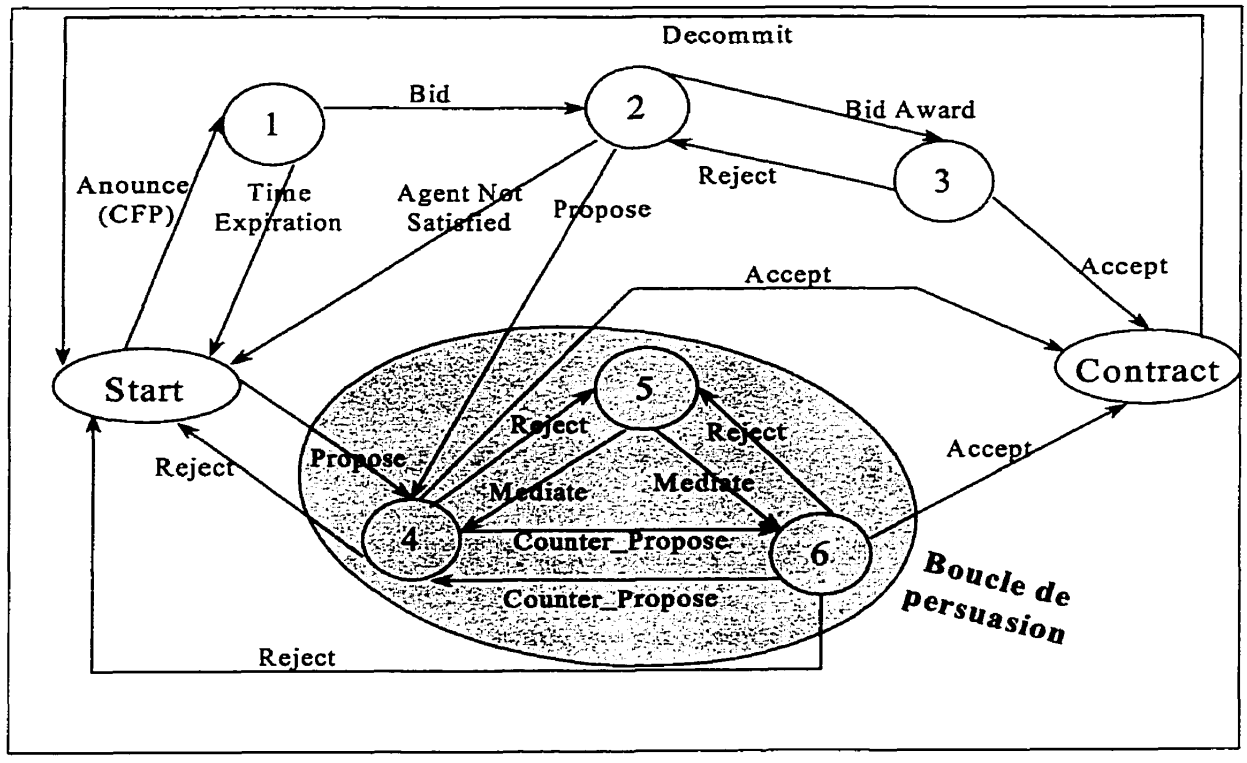


Figure 5.2: Diagramme de transition d'états de la négociation

Ainsi, le modèle que nous proposons est articulé autour de trois principales composantes : le processus de négociation, le processus de persuasion et la stratégie de négociation.

Le processus de négociation est organisé en deux phases : une phase pour la négociation coopérative et une autre pour la négociation persuasive. La section 5.2 présente en détail un tel processus. Afin d'améliorer l'efficacité des agents en terme de profit et d'accélérer la convergence du processus de négociation, différentes techniques ont été introduites dans le modèle : (1) différents types de contrats et (2) stratégie basée sur l'évaluation du risque. La stratégie de négociation définit le processus selon lequel la négociation sera conduite. Elle est le moteur du processus de négociation. La section 5.4 présente en détail la stratégie de négociation.

Le processus de persuasion que nous proposons utilise plusieurs mécanismes d'argumentation : (1) normes sociales, (2) génération de promesses ou menaces en

exploitant les relations de dépendances, (3) argumentation par rappel de contre-exemple, et (4) génération de solutions en utilisant un raisonnement par cas. La section 5.3 présente en détail le processus d'argumentation.

5.1 Principes utilisés pour l'élaboration du modèle

Ces principes rapprochent le développeur du cadre d'application et des considérations à prendre lors de l'implantation de ce modèle. Ces principes de travail ont été définis dans le but de simplifier les contraintes auxquelles nous devrons nous soumettre pour l'élaboration du modèle de coopération et afin de chercher plus de réalisme que de perfectionnisme. Ainsi, les paragraphes suivants présentent les principes que nous avons adoptés dans le cadre de ce travail avec une brève justification de leurs raisons d'être.

1- le principe de la non-bienveillance *:* les agents ne sont pas, à priori, censés s'aider les uns les autres. Ils décident de façon autonome s'ils acceptent de coopérer ou non avec les autres, en s'appuyant sur la valeur d'utilité générée.

raisons : d'un point de vue modélisation cognitive, cela introduit un cadre théorique suffisamment riche pour permettre d'analyser et prédire l'occurrence de plusieurs types d'interactions sociales significatives, dont la coopération bienveillante n'est qu'un cas particulier (où l'utilité personnelle de chaque agent est nulle). Cela permet aussi d'étendre notre modèle pour d'autres applications où les agents sont concurrents ou travaillent pour leur intérêt personnel (commerce électronique, ingénierie concurrente,...). En particulier, un tel principe nous permet de prendre en compte les notions d'adoption de buts et de persuasion, absentes dans les approches fondées sur le principe de bienveillance. Par ailleurs, il nous fournit un moyen d'examiner la dynamique du pouvoir social des agents appartenant à une société, lorsque certains y entrent ou bien en sortent dynamiquement.

2- le principe de la sincérité *:* les agents ne choisissent jamais, de façon délibérée, de donner une information incorrecte aux autres avec le but de les exploiter. Ils ne communiquent aux autres que ce qu'ils croient eux-mêmes.

raisons : de manière générale, le raisonnement sur les autres agents et la révision des croyances sont des traitements imbriqués dans notre modèle : un agent utilise ses informations sur les autres pour interagir socialement et de sa part les résultats de cette interaction (même si non réussie) lui permettent de mettre à jour ces informations. Ainsi, on trouve dans la mise à jour de la base de connaissance d'un agent une certaine naïveté vis à vis de ses pairs, et pour ne pas encombrer le modèle par une gestion de la notion de crédibilité chez les agents (notre but principal est d'explorer la notion de persuasion à base des relations de dépendances) nous avons opté pour le principe de sincérité chez les agents.

3- le principe de l'auto-connaissance *:* les agents ont une représentation correcte et complète relative à leurs propres propriétés, c'est à dire, ils connaissent leurs buts, leurs capacités, leurs croyances et leurs engagements. En ce qui concerne leur représentation des autres, les agents n'ont que des modèles qui peuvent être erronés.

raisons : un tel principe est nécessaire pour la révision de croyances. En effet, l'adoption de nouvelles croyances est une décision que l'agent prend en fonction de plusieurs critères : (1) la cohérence avec les croyances qu'il possède, et (2) la crédibilité et les compétences de la source. Pour le premier critère, un agent accepte une croyance s'il peut la justifier ou au moins l'insérer sans inconsistance dans sa base, et pour le deuxième critère, la crédibilité est garantie par le 2^{ieme} principe (principe de sincérité), mais reste à résoudre le problème de la compétence. Pour cela, nous avons introduit le principe d'auto-connaissance qui définit que chaque agent est expert de ce qui le concerne. Ainsi, un agent accepte les informations fournies par un agent sur lui-même. Ce principe peut être exprimé formellement par :

soit *X, Y* deux agents, *P* un but, et '*Bel*' l'opérateur modal de croyance ("Belief."). Selon le principe d'accepter une connaissance qui vient d'un expert, comme une vérité on a:

$Bel_X(expert(Y\ P)) = Bel_X(\ Bel_Y(P)) \rightarrow Bel_X(P\ )$

et

$Bel_X(Goal(X\ P)) \rightarrow Goal(X\ P)$

Si on n'avait pas introduit explicitement que chaque agent n'est expert que de ce qui le concerne, on pourrait alors trouver chez les agents des erreurs d'inférence du type suivant :

$\{ Bel_X(expert(Y\ P))\ et\ Bel_X(Bel_Y(Goal(X\ P))) \} \rightarrow Bel_X(Goal(X\ P))$

et puisque $Bel_X(Goal(X\ P)) \rightarrow Goal(X\ P)$. Alors on se trouve avec une situation où le but de X est à la merci de la croyance de Y ! !.

4- le principe de la cohérence : les agents n'ont pas de croyances contradictoires sur les autres. Si une inconsistance est détectée, les agents révisent leurs croyances sur les autres afin de rétablir la consistance.

raisons : le modèle de formation de coalition que nous utilisons suppose implicitement que les agents exploitent une représentation qui leur est interne ou à laquelle ils peuvent accéder concernant certaines propriétés des autres (les modèles des autres ne sont supposés ni complets ni corrects). Puisque notre objectif est d'étudier le comportement social des agents dans un contexte de SMA ouvert où cette représentation des autres doit être mise à jour de façon dynamique, et puisque dans les mécanismes de raisonnement des agents nous n'utilisons qu'une logique de premier ordre augmentée par des opérateurs modaux qui ne supportent pas le raisonnement flou ou incertain, alors le principe de la consistance s'avère nécessaire comme hypothèse de travail.

5- le principe de l'autonomie limitée : ce qui limite l'autonomie des agents c'est leur évolution dans un même environnement ; cela entraîne une interférence entre leurs actions de façon négative ou positive et leur compétition pour les ressources.

Une relation d'interférence (dépendance) positive existe entre deux agents X et Y si : (1) X a pour but P, et (2) Y fait une certaine action qui implique P.

- Une relation d'interférence négative existe entre X et Y si : (1) X a pour but P, et (2) Y fait une certaine action qui implique non P,
- Une relation de compétition existe entre X et Y sur une ressource R si : (1) tous les deux ont besoin de la ressource R, et (2) la ressource R ne peut pas être utilisée simultanément par les deux agents.

5.2 Processus de négociation

La procédure de négociation que nous proposons est un processus itératif d'interaction de trois tâches principales : la génération de proposition, la génération de contre-proposition ou de feed-back, et la génération d'argumentation. En se basant sur les fonctions d'utilités et l'évaluation du risque, notre stratégie de négociation (cf. 5.4) utilise un algorithme itératif qui peut être arrêté en tout temps t_i en produisant comme résultat la meilleure solution qui puisse être atteinte à ce temps t_i, sans avoir besoin de faire des retours arrière ni de prospecter le futur.

La procédure de négociation que nous proposons est divisée en deux phases qui peuvent être considérées de façon indépendante ou complémentaire :

Négociation coopérative : cette phase sert à déterminer les partenaires potentiels et éventuellement conclure un contrat. Elle utilise plusieurs techniques pour la construction des propositions (bases de cas, fonctions d'utilités, graphe de dépendances). Après avoir sélectionné le but à satisfaire, l'agent se trouve dans l'une des situations suivantes : (1) il connaît un partenaire potentiel ; dans ce cas, il s'engage directement dans une phase de négociation persuasive avec ce partenaire. (2) il ne connaît aucun partenaire potentiel ; dans ce cas, il utilise l'un des protocoles de négociation (Contract-net cf.4.7, FIPA-Contract-net[30], enchère anglaise cf. 4.2, enchère hollandaise cf. 4.2, enchère de Yankee cf. 4.2) pour recevoir des offres. Ensuite, selon l'évaluation des offres, cette phase peut conclure, soit à un contrat si l'une des offres est satisfaisante, soit à l'identification d'un partenaire potentiel avec qui la deuxième phase s'engage, soit à un échec du but si le temps limite est atteint sans succès. La figure 5.3 présente l'algorithme général de la négociation coopérative.

Les figures suivantes qui décrivent le processus de négociation (figure 5.3 et figure 5.4) et de persuasion (figure 5.5) doivent être lues de haut en bas. Les rectangles à coins arrondis représentent les procédures utilisées dans un traitement. Les ellipses représentent

[30] FIPA-Contract-net : est une version du Contract-net dans laquelle est ajoutée la possibilité de faire plusieurs itérations. En effet, le manager fait son annonce, puis reçoit des propositions qu'il évalue et par la suite il peut soit accepter des offres, rejeter des autres, ou s'engager dans une autre itération en envoyant une autre annonce modifiée.

les connaissances ou structures de données utilisées dans le traitement. Les rectangles en coins réguliers représentent les traitements effectués à chaque étape ; ceux en double ligne désignent les étapes début et fin du processus. Finalement, les losanges représentent les points de décision dans le processus.

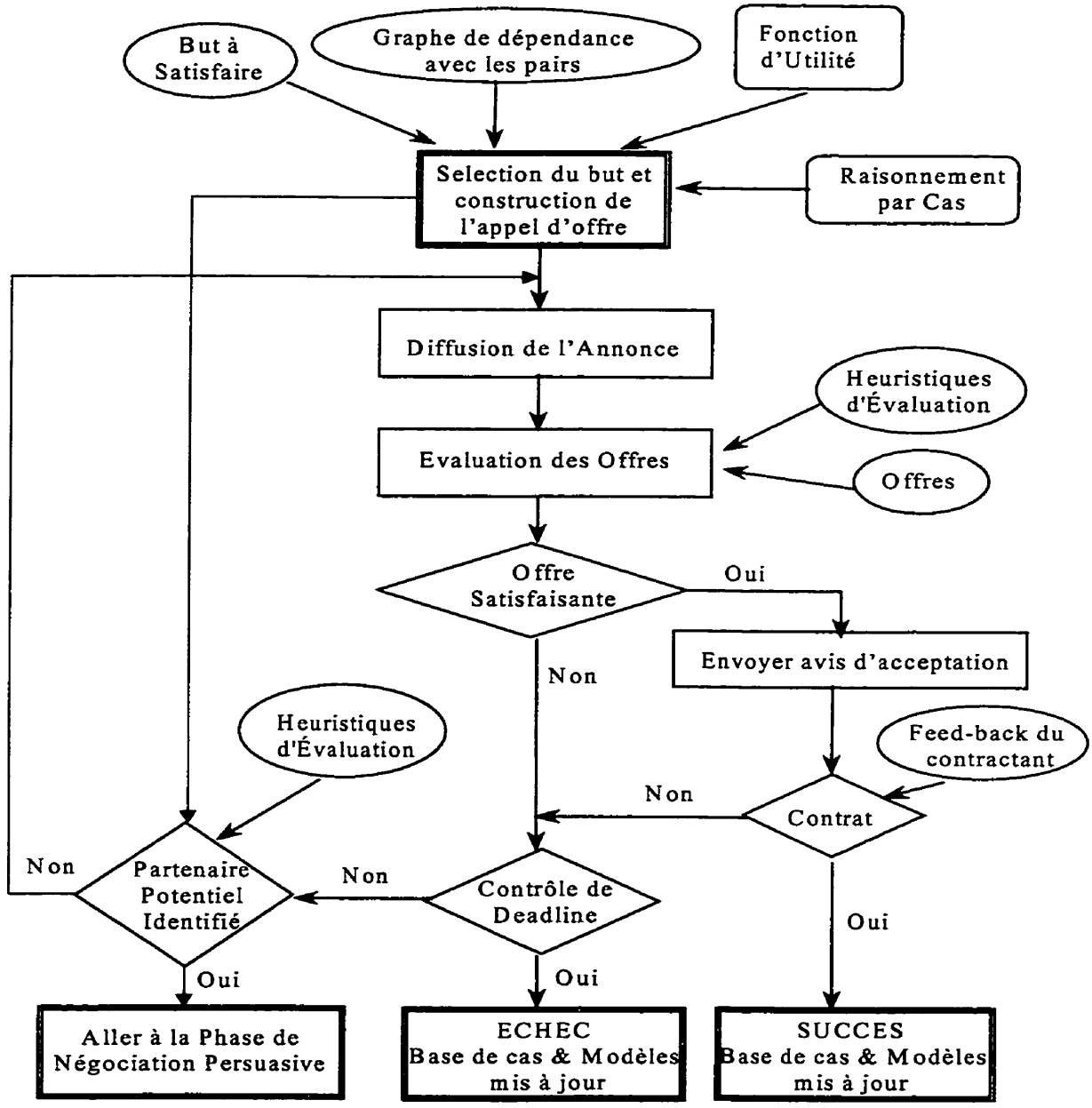


Figure 5.3: Processus de négociation coopérative

Négociation persuasive : cette deuxième phase permet d'entrer dans un cycle de négociation basé sur l'argumentation où les agents essayent de convaincre les uns les autres de changer leurs attitudes envers les propositions faites, et ce en générant des messages d'argumentation ou des contre propositions argumentées. Cette phase se déroule selon un processus itératif comprenant plusieurs traitements. D'abord, l'agent génère sa proposition en utilisant ses fonctions d'utilité, un raisonnement par cas, et les graphes de dépendances. Ensuite, la proposition générée est évaluée par rapport aux cas d'échecs pour éviter un rejet potentiel. Après la soumission de la proposition, si elle est acceptée, l'agent conclut un contrat et arrête la négociation ; sinon il s'engage dans un processus de persuasion pour essayer de convaincre son partenaire potentiel. Si la persuasion n'est pas possible, il essaie de modifier sa proposition ou même générer une nouvelle puis répéter une autre itération.

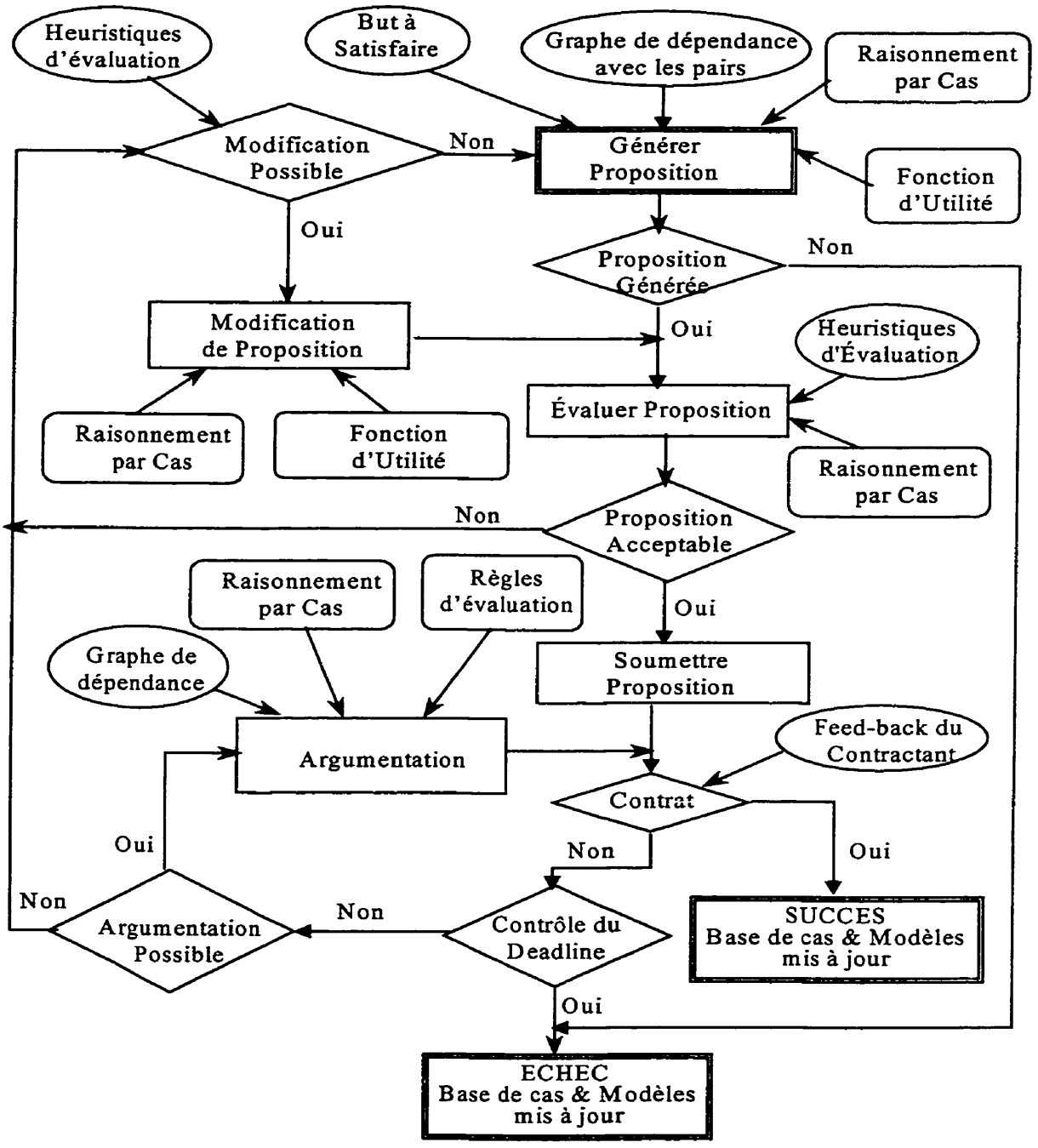


Figure 5.4: Processus de négociation persuasive

Dans le processus de négociation décrit ci-dessus, on peut facilement voir que les trois principales composantes (fonctions d'utilité, raisonnement par cas et graphe de dépendances) sur lesquelles repose notre modèle sont impliquées dans les trois

principales tâches de la négociation : la génération de la proposition initiale, l'argumentation et l'amélioration de proposition rejetée.

Les fonctions d'utilité (cf. Annexe I) sont la clé aussi bien pour la construction de propositions que pour faire un choix parmi les propositions reçues. En fait, chaque proposition est évaluée en terme d'un nombre d'attributs considérés par l'agent, ce qui fait que la fonction d'utilité permet d'identifier la structure de préférence de l'agent. La valeur d'utilité d'une proposition est évaluée sur un ensemble d'attributs $T=\{t_1, t_2, ..., t_n\}$ comme suit :

$U(t_1,t_2, ..., t_n) = f[V_1(t_1), V_2(t_2), ..., V_n(t_n)] = w_1 * V_1(t_1) + w_2 * V_2(t_2) + ... + w_n * V_n(t_n)$,

où wi mesure l'importance que l'agent accorde à chaque attribut, et $V_i(t_i)$ la valeur de l'attribut t_i.

Dans la section cf. 5.4, nous définissons la stratégie de négociation basée sur les fonctions d'utilités et l'évaluation du risque que les agents utilisent pour construire, améliorer, accepter ou refuser une offre. En fait, cette stratégie constitue le moteur de la négociation où les agents s'échangent des messages de proposition et contre-proposition, conduisant soit à l'élaboration de contrat, soit à une situation de conflit qui nécessite un processus de persuasion.

Le raisonnement par cas (CBR) [Kolodner 1989] est un processus de résolution basé sur l'utilisation de solutions passées pour traiter de nouveaux problèmes. En fait le CBR consiste à réutiliser les cas de succès du passé et à éviter ses échecs. La mémoire des cas contient aussi bien les cas de succès que ceux d'échecs. Les cas de succès sont utilisés pour la construction de propositions alors que les cas d'échecs sont utilisés pour l'évaluation de propositions en vue d'éviter les rejets potentiels.

Dans ce modèle, les agents utilisent le raisonnement par cas pour faire face aux situations d'interactions non familières que nous avons identifiées dans la section cf. 3.2.2. En effet, la négociation est un processus complexe qui nécessite plusieurs étapes. Ainsi, en utilisant le CBR, l'agent aura un moyen pour apprendre des plans de résolution complexes qu'il pourra réutiliser au lieu de refaire des plans pour les mêmes buts qui ont été déjà planifiés. Il n'existe pas de modèle suffisamment général pour la négociation. De

ce fait, la réutilisation de plans ayant déjà donné de bons résultats dans des situations similaires est d'un grand intérêt.

5.3 Processus de persuasion

Le processus de persuasion se définit comme la manière avec laquelle un agent raisonne sur un autre agent en utilisant un modèle de ce dernier, pour déterminer autant de façon que le modèle le permette pour influencer le comportement de celui ci [Aubuchon 1997]. L'objectif de la génération d'arguments de persuasion est soit de changer la structure des croyances de l'agent envers la proposition ou de changer le coefficient d'importance accordé à la cause du rejet.

Deux méthodes sont généralement utilisées pour générer une argumentation : (1) construire des arguments à partir des données de base (croyances, graphe de dépendances avec les pairs, règles définissant les normes sociales) ou (2) utiliser le CBR pour rappeler des cas de propositions acceptées dans le passé pour des situations similaires à celle en cours. La persuasion à base de CBR avait déjà été utilisée avec succès par Sycara pour la construction de son '*PERSUADER*' [Sycara 1985, Sycara 1988b, Sycara 1993], un système d'aide à la résolution de conflits dans le domaine de travail syndiqué (Syndicat vs Patron). Nous avons adapté les algorithmes d'argumentation à base de CBR proposés par Sycara à notre contexte. Cependant, dans l'implantation de notre système (*MIAMAP* cf. 6), cette partie n'est pas encore active et pour l'expérimentation (cf. 7) nous avons seulement évalué la persuasion à base des relations de dépendances.

Afin de générer des arguments efficaces, l'agent argumentant a besoin de connaître aussi bien la structure de croyances de son partenaire que l'influence de l'environnement sur cette structure. Ainsi pendant la négociation, les réactions de l'autre agent servent à mettre à jour le modèle que maintient l'argumentant de son partenaire.

Cinq principaux types d'arguments ont été identifiés pour une négociation :

- persuader l'agent en lui rappelant une loi sociale qu'il a violée si c'est le cas,

- persuader l'agent en pointant une incohérence de celui-ci avec son passé. Cela est réalisé en lui rappelant un contre exemple de l'historique de son passé dans lequel il a accepté une valeur inférieure ou égale à celle offerte,
- persuader l'agent en pointant une incohérence de celui-ci avec son environnement. Cela est réalisé en lui indiquant une offre dont la valeur est inférieure ou égale à celle offerte, et qui a été acceptée par un des agents qui lui est similaire,
- persuader l'agent par des promesses. Cela est accompli en promettant à l'agent la réalisation de buts pour lesquels il dépend mutuellement ou unilatéralement (cf. 3.2.3) du persuadeur,
- persuader l'agent par la génération de menaces en exploitant des dépendances mutuelles ou unilatérales sur des buts.

En présence de plusieurs arguments, l'agent argumentant utilise une stratégie qui consiste à présenter celui ayant la plus faible puissance en premier.

La figure 5.5 présente le processus de gestion des conflits et de génération d'arguments.

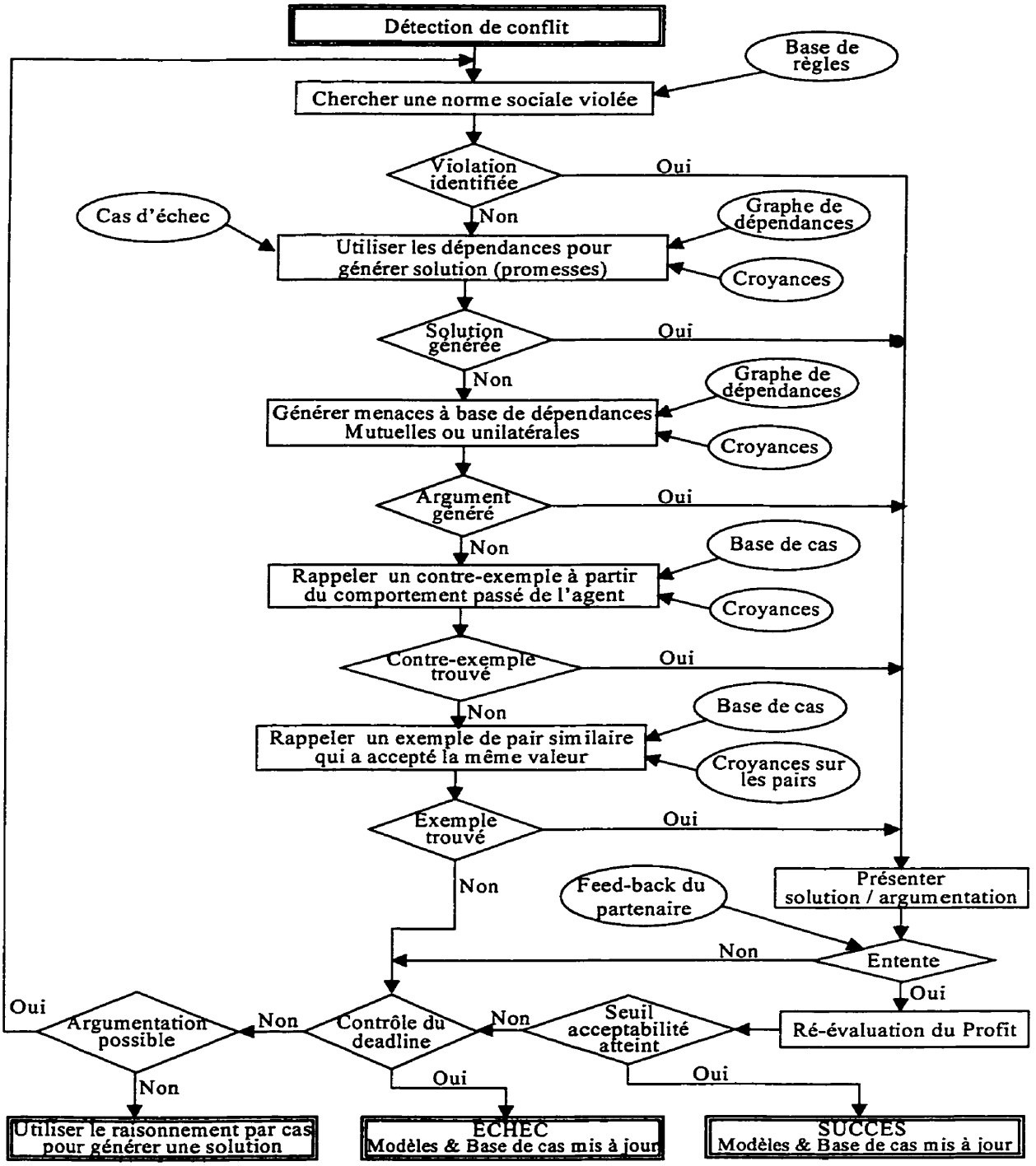


Figure 5.5: Processus de persuasion

L'ordre d'utilisation des différents mécanismes pour la résolution de conflit tel que présenté dans la figure 5.5, a été fait selon notre intuition de facilité de génération de solutions ou d'arguments. En fait, la stratégie citée plus haut consistant à présenter en premier l'argument ayant la plus faible puissance n'est pas respectée, car les métriques pour mesurer la puissance des arguments n'ont pas encore été établies.

5.3.1 Génération d'arguments basés sur les normes sociales

Les normes constituent le lien entre le raisonnement individuel de l'agent et le raisonnement social de la collectivité. Ainsi, dans une société d'agents cognitifs nous avons à considérer, d'une part, le caractère prescriptif des normes sociales, c'est à dire leur rôle à contrôler et régulariser le comportement des agents (émergence d'un comportement social), et d'autre part, le caractère mental des normes qui permet à des agents cognitifs rationnels de devenir aussi des agents normatifs (c'est à dire des agents qui veillent au respect des normes dans la société).

Les lois sociales font partie intégrante des croyances des agents et peuvent être représentées de deux façons : sous formes de contraintes (règles de contrôle), ou sous formes d'objets de l'état mental de l'agent (engagements à satisfaire).

Pendant la négociation, chaque proposition est évaluée non seulement du point de vue profit ou satisfaction du but objet de l'interaction, mais elle doit aussi répondre aux normes sociales définies pour le SMA. Ainsi, la violation d'une norme sociale constitue un très bon argument pour convaincre l'autre agent de modifier sa proposition.

5.3.2 Persuasion à base des relations de dépendances

Le processus de génération de solution à base des relations de dépendances, exploite le graphe de dépendances pour générer plusieurs types de contrats. Avant de présenter ce processus nous allons d'abord introduire les différents types de contrats générés tout en montrant leur intérêt dans la résolution de conflits.

Contrat par échange de buts : parfois, il est intéressant de faire un échange de buts entre les agents pour améliorer la solution globale. Plus particulièrement, les contrats par échange de buts deviennent plus intéressants avec les agents ayant des relations de

dépendances mutuelles (cf. 3.2.3). Dans un contrat avec échange de buts, le premier agent sous-traite la réalisation de son but auprès de l'autre agent, tandis que l'autre agent sous-traite, en contrepartie, un de ses buts. Une forme plus générale des contrats avec échange de buts est celle qui fait intervenir plus de deux buts. Le graphe de dépendances est d'une grande utilité pour l'implantation d'un tel mécanisme. D'une part, il permet de déterminer si un échange de buts est bénéfique pour la solution globale et, d'autre part, il rend l'identification d'échange triviale. Ce type de contrat permet de dénouer l'impasse causée par un conflit sur l'utilité des agents en faisant des promesses (au sens d'engagements) de réalisation de buts en contre partie du manque à gagner. L'échange est fait de façon à ce que les gains ou pertes des agents soit équivalents.

Contrat par regroupement de buts *:* ce type de contrat est construit en regroupant un sous-ensemble de buts dans le même contrat, de manière à rendre le coût global du contrat profitable pour les deux agents, même si le coût individuel relatif à quelques buts peut être supérieur au coût acceptable. Les contrats par regroupement de buts sont intéressants avec des agents ayant des relations de dépendances multiples (cf. 3.2.3), et deviennent plus intéressants si on se rappelle que, dans la vie réelle, le coût de réalisation d'un ensemble de buts est généralement inférieur à la somme des coûts individuels des buts.

Contrat multiagent *:* la construction de contrat multiagent consiste à impliquer plusieurs agents dans un même contrat de façon que le coût de réalisation global soit profitable pour tous les agents. Un tel contrat peut aussi comprendre des échanges ou des regroupements de buts. L'implantation d'un tel mécanisme nécessite de faire circuler le message du contrat sur toutes les parties et le contrat devient valide si tous les agents l'acceptent. Dans notre modèle, la construction d'un tel type de contrat se fait en utilisant le graphe de dépendances. En effet, on parcourt le graphe de dépendances à la recherche d'un cycle balancé entre les deux agents en conflit, et l'ensemble des transactions constituant le cycle déterminera la répartition des buts entre les agents impliqués dans le contrat multiagent qui en découle.

Dans le processus de génération de solution à base des relations de dépendances, l'agent cherche d'abord à exploiter les dépendances mutuelles pour construire un contrat avec échange de buts. Si aucune dépendance mutuelle n'est trouvée, l'agent essaie de construire un contrat par groupement en exploitant des relations de multi-dépendances envers son pair. Enfin, s'il n'y a pas de multi-dépendances, l'agent cherche un cycle dans le graphe de dépendances entre lui et son pair pour construire un contrat multiagent.

L'algorithme général pour la génération de solution à base de dépendances se déroule comme suit :

étape1 : construire et proposer un contrat par échange de buts à base de dépendances mutuelles,

1.1 chercher le graphe pour toutes les dépendances mutuelles existantes entre les deux agents,

1.2 trier ces dépendances selon leur force (cf. 3.2.3),

1.3 choisir la première dépendance,

1.4 construire et proposer un contrat par échange de buts,

1.5 si le contrat est accepté alors succès, fin.

1.6 sinon sélectionner la prochaine dépendance mutuelle, aller à l'étape 1.4,

1.7 si l'ensemble des dépendances mutuelles est épuisé sans succès alors aller à l'étape 2,

étape2 : construire et proposer un contrat par regroupement de buts à base de multi-dépendances,

2.1 chercher le graphe pour toutes les multi-dépendances de l'agent envers son pair,

2.2 trier ces dépendances selon leur force,

2.3 choisir la première dépendance,

2.4 construire et proposer un contrat par regroupement de buts,

2.5 si le contrat est accepté alors succès, fin.

2.6 sinon sélectionner la prochaine multi-dépendance, aller à l'étape 2.4,

2.7 si l'ensemble des multi-dépendances est épuisé sans succès alors aller à l'étape 3,

étape3 : construire et proposer un contrat multiagent à base des cycles de dépendances,

3.1 chercher le graphe pour un cycle balancé de dépendances constituant un chemin[31] fermé entre les deux agents,

3.2 construire un contrat multiagent en utilisant le cycle de dépendances identifié,

3.3 proposer le contrat multiagent au groupe d'agents impliqués dans le contrat,

3.4 si tous les agents acceptent le contrat alors succès, fin.

3.5 sinon aller à l'étape 3.1,

3.6 s'il n'existe plus de cycle de dépendances entre les deux agents alors aller à l'étape 4,

étape4 : utiliser le raisonnement par cas pour générer une solution (cf. 5.3.5).

5.3.3 Argumentation par la génération de menaces

Quand il n'y a pas de concessions sur les buts en conflit, des menaces sont générées pour induire une telle concession. La procédure suivante décrit les étapes pour la génération de menaces : Soit A_1 l'agent argumentant, A_2 l'agent à convaincre, X le but ou la tâche qui cause le conflit, et V_1 le profit associé à X pour A_1.

1. trouver un but de A_2 contrôlée par A_1, qui permet de compenser la perte causée pour A_1 par l'insatisfaction de son but. Ceci est réalisé en parcourant le graphe de dépendances, en prenant comme point de départ A_2, à la recherche d'une chaîne de dépendance unilatérale forte de A_2 vers A_1. La chaîne doit satisfaire la condition que le coût des buts impliqués dedans soit supérieur ou égal à V_1,
2. propager l'effet négatif de la chaîne de dépendance unilatérale sur tous les agents impliqués le long du chemin en proposant un contrat multiagent qui comprend les concessions nécessaires,
3. examiner tous les chemins partant de A_2, jusqu'à ce qu'une chaîne de dépendance unilatérale soit trouvée ou tous les chemins soient examinés.

[31] Un chemin d'un nœud X vers un nœud Y dans le graphe constitue une chaîne causale qui permet de déterminer la valeur des transactions qui doit être balancée dans l'autre sens (de Y vers X) pour construire un cycle.

5.3.4 Génération d'arguments par rappel de contre exemple

Ce type d'argumentation est basé sur le rappel au partenaire d'une contradiction avec son passé ou son environnement en utilisant la base de cas. La procédure pour générer une telle argumentation se résume comme suit :

a- rappeler un contre-exemple de l'historique du comportement du partenaire à convaincre, selon la procédure suivante :

1. trouver dans la base de cas, un contrat pour lequel l'agent avait accepté une valeur inférieure ou égale à celle de l'offre en cours,
2. présenter le cas comme une contradiction de l'agent avec son passé,
3. si cas non trouvé, aller à l'étape b.,

b- rappeler à l'agent un exemple de cas qui a été accepté par l'un des agents qui lui sont similaires, selon la procédure suivante :

1. trouver des cas similaires dans le comportement des pairs de l'agent à convaincre,
2. sélectionner, parmi ces cas, ceux dont la valeur du but est inférieure à la valeur demandée,
3. présenter ces cas à l'agent à convaincre.

5.3.5 Génération de solution à base de CBR

Avec le message de rejet de l'offre, l'agent obtient les informations sur la composante causant le rejet, l'importance de celle-ci, et la raison du rejet. En utilisant comme indices les caractéristiques de la composante et la raison du rejet, l'agent recherche dans sa base les cas similaires à la situation en cours. Si un cas similaire est trouvé, l'agent le modifie pour augmenter son adéquation[32] à la situation présente.

L'algorithme général de la procédure de raisonnement par cas se présente comme suit :

1. trouver les cas similaires dans la base de cas en utilisant les caractéristiques de la situation présente et des métriques de similarité[33],

---

[32] Le critère utilisé pour mesurer l'adéquation d'une solution est qu'elle doit augmenter le profit de l'agent à convaincre plus qu'elle ne réduit le profit de celui qui la propose.

[33] Le degré de similarité dépend de plusieurs critères : l'appartenance au même niveau dans la hiérarchie des cas, le nombre d'attributs en commun, la différence d'échelle entre les deux cas, la différence dans les coûts unitaires. Ainsi que d'autres critères dépendant du domaine.

2. trier les cas trouvés en fonction de leur appropriation à la négociation en cours,
3. sélectionner le meilleur cas de la liste triée,
4. construire une proposition : (a) extraire les connaissances pertinentes, (b) ajuster les connaissances extraites en utilisant l'état mental et les buts de l'agent, (c) construire la proposition,
5. évaluer la proposition construite en utilisant les cas d'échecs en mémoire et la fonction d'utilité,
6. si nécessaire, modifier la proposition construite, puis la soumettre au partenaire,
7. si la proposition est acceptée, alors stop. Sinon aller à l'étape3,
8. si l'ensemble des cas extraits est épuisé sans succès alors envoyer un message pour arrêter la négociation et mettre à jour la base de cas et les croyances sur le partenaire.

5.4 Stratégie de négociation utilisée par les agents

La stratégie de négociation constitue le moteur des négociations, aussi bien pour la phase coopérative que pour la phase persuasive (cf. 5.2). En fait, puisque les agents sont supposés individuellement rationnels[34], il est clair que si aucun des agents ne fait de concession pendant une étape du cycle de négociation, les deux agents passent dans une situation de conflit. Une telle situation peut se produire même si l'espace des contrats possibles entre les deux agents est non vide.

Une bonne façon pour déterminer quel agent doit faire une concession à chaque étape est de considérer le manque à gagner que va subir chacun des agents pendant cette étape si la négociation s'arrête sur une situation de conflit. En fait, le principe de la stratégie que nous proposons repose sur le fait que «plus l'agent a déjà fait des concessions dans le passé, moins se sera à lui de faire la prochaine concession».

Ainsi, on utilise le critère d'évaluation de risque proposé par Zeuthen [Zeuthen 1930] pour déterminer quel agent doit faire une concession la prochaine étape. Plus précisément nous avons adapté la stratégie de Zeuthen [Rosenschein 1994] pour en construire une qui

[34] Un agent est dit individuellement rationnel s'il n'accepte que des contrats qui lui permettent de réaliser un profit positif.

permet de supporter l'incertain et le manque de connaissances[35] que l'agent a envers son pair. Dans cette adaptation, chaque agent utilise un modèle de son partenaire pour estimer l'offre minimale acceptée par son pair. En effet, si à une étape *t*, l'agent décide de ne pas faire de concession, il prend donc le risque que son partenaire ne fasse pas lui aussi de concession et que la négociation s'arrête sur une situation de conflit. La stratégie peut donc être exprimée de façon formelle comme suit :

Soit deux agents, A_i ayant un but G_i à satisfaire et A_j son partenaire, ainsi que les variables suivantes :

γ_i : le coût maximal acceptable pour A_i

γ_j : le prix minimal acceptable pour A_j

$\dot{\gamma}_m$: l'estimation par A_i du prix minimal acceptable pour A_j

$\dot{\gamma}_m$: l'estimation par A_j du coût maximal acceptable pour A_i

Du point de vue de A_i, l'utilité et le risque sont définis par:

l'utilité que l'agent A_i croit obtenir en faisant une offre de valeur $\delta_i(t)$ est:

$U_i^i(\delta_i(t)) = max\ (\gamma_i - \delta_i(t),\ 0)$

l'utilité que l'agent A_i croit obtenir en acceptant l'offre $\delta_j(t)$ de A_j est:

$U_i^i(\delta_j(t)) = max\ (\gamma_i - \delta_j(t),\ 0)$

l'utilité que l'agent A_i croit que l'agent A_j va obtenir en faisant une offre de valeur $\delta_j(t)$ est approximée par: $U_i^j(\delta_j(t)) = max\ (\delta_j(t) - \dot{\gamma}_m,\ 0)$

l'utilité que l'agent A_i croit que l'agent A_j va obtenir en acceptant l'offre $\delta_i(t)$ de A_i est approximée par: $U_i^j(\delta_i(t)) = max\ (\delta_i(t) - \dot{\gamma}_m,\ 0)$

On peut calculer le risque que chacun des agents est prêt à prendre, s'il ne fait pas de concession et ainsi, laisser la négociation atteindre une situation de conflit.

Pour l'agent A_i le risque qu'il est prêt à prendre est:

$$Risk_i^i(t) = \begin{cases} 1 & \text{Si } U_i^i(\delta_i(t)) = 0 \\ (U_i^i(\delta_i(t)) - U_i^i(\delta_j(t))) / U_i^i(\delta_i(t)) & \end{cases}$$

[35] Dans la stratégie Zeuthen, il est supposé que les agents possèdent des connaissances complètes et parfaites sur leurs pairs, chose que nous croyons irréaliste à supposer et à implanter dans un SMA.

le risque que l'agent A_i croit que l'agent A_j est prêt à prendre est:

$$Risk_i^j(t) = \begin{cases} 1 & \text{Si } U_i^j(\delta_j(t)) = 0 \\ (U_i^j(\delta_j(t)) - U_i^j(\delta_i(t))) / U_i^j(\delta_j(t)) \end{cases}$$

Du point de vue de A_j, l'utilité et le risque sont définis par:

l'utilité que l'agent A_j croit obtenir en faisant une offre de valeur $\delta_j(t)$ est:

$U_j^j(\delta_j(t)) = max\ (\delta_j(t)-\gamma_j,\ 0)$

l'utilité que l'agent A_j croit obtenir en acceptant l'offre $\delta_i(t)$ de A_i est:

$U_j^j(\delta_i(t)) = max\ (\delta_i(t)-\gamma_j,\ 0)$

l'utilité que l'agent A_j croit que l'agent A_i va obtenir en faisant une offre de valeur $\delta_i(t)$ est approximée par: $U_j^i(\delta_i(t)) = max\ (\gamma_m^i-\delta_i(t),\ 0)$

l'utilité que l'agent A_j croit que l'agent A_i va obtenir en acceptant l'offre $\delta_j(t)$ de A_j est approximée par: $U_j^i(\delta_j(t)) = max\ (\gamma_m^i-\delta_j(t),\ 0)$

On peut calculer le risque que chacun des agents est prêt à prendre, s'il ne fait pas de concession et ainsi, laisser la négociation atteindre une situation de conflit.

Pour l'agent A_j le risque qu'il est prêt à prendre est:

$$Risk_j^j(t) = \begin{cases} 1 & \text{Si } U_j^j(\delta_j(t)) = 0 \\ (U_j^j(\delta_j(t)) - U_j^j(\delta_i(t))) / U_j^j(\delta_j(t)) \end{cases}$$

le risque que l'agent A_i croit que l'agent A_j est prêt à prendre est:

$$Risk_j^i(t) = \begin{cases} 1 & \text{Si } U_j^i(\delta_i(t)) = 0 \\ (U_j^i(\delta_i(t)) - U_j^i(\delta_j(t))) / U_j^i(\delta_i(t)) \end{cases}$$

Pour chaque agent, $Risk_i^i(t)$ donne une indication sur la volonté de l'agent A_i de risquer un conflit. Ainsi, plus la valeur du $Risk_i^i(t)$ augmente, plus l'intérêt de A_i à faire une concession diminue. Intuitivement, on propose une stratégie où l'agent qui a le plus petit risque fera une concession à la prochaine étape.

À chaque étape t, chaque agent calcule son risque, $Risk_i^i(t)$, et estime celui de son partenaire $Risk_i^j(t)$. Si son risque est inférieur à celui de son partenaire, il doit alors proposer une offre avec la concession minimale[36] suffisante[37] pour faire basculer le risque de son partenaire. Toutefois, à chaque étape les agents ne proposent que des offres qui leur procurent une utilité non nulle.

5.4.1 Évaluation de la stratégie de négociation

L'évaluation de la stratégie de négociation peut être faite selon plusieurs critères [Fudenberg 1991, Rosenschein 1994]: l'efficacité sur le profit, le bien être social, la stabilité, la simplicité, l'efficacité computationnelle et l'efficacité dans la communication. Dans ce paragraphe, nous examinons la stratégie sur trois de ces critères : l'efficacité sur le profit, la stabilité et la simplicité. Les autres critères seront traités dans le chapitre 7 avec des résultats expérimentaux.

a- efficacité de la stratégie : deux agents qui utilisent la stratégie de négociation décrite ci-dessous vont toujours conclure un contrat s'il existe une entente acceptable pour les deux agents et ce contrat maximise le produit de leur utilité. En effet, le fait que les deux agents arrivent toujours à un contrat s'il en existe un, est garanti par le fait qu'à chaque étape au moins un des agents doit faire une concession. Cependant, pour qu'un tel contrat soit optimal, il faut qu'il maximise le produit et la somme de leurs utilités. La maximisation du produit garantit l'optimalité individuelle et la maximisation de la somme garantit l'optimalité du bien être social[38](profit conjoint).

Lemme 1 *: les offres faites par les agents dans différentes étapes d'une négociation suivent une série monotone croissante.*

Soit deux agents A_1 et A_2, utilisant tous deux la stratégie de négociation proposée ci-dessous, et considérant les variables suivantes :

$\delta_i(t)$: l'offre de l'agent A_i à l'étape t.

---

[36] Une concession minimale suffisante est une concession suffisante que l'agent croit apte à procurer le minimum nécessaire d'utilité à son partenaire.

[37] Une concession suffisante est une concession qui permet de changer la balance des risques entre l'agent et son partenaire. Par exemple, après une concession suffisante faite par A_i on aura $Risk_i^i(t) > Risk_i^j(t)$.

[38] Le bien être social : définit le profit global du système, c'est à dire la somme des profits individuels des composantes du système.

$\omega_i(t)$: la concession faite par l'agent A_i à l'étape t.

Sur un ensemble d'étapes de la négociation $\{t_1, t_2, ..., t_n\}$, la concession globale pour chacun des agents est déterminée par la somme $\Omega(t_n) = \Sigma^n_{i=1}\ \omega(t_i)$.

Selon le protocole que nous utilisons, à chaque étape t_i de la négociation, seul un des agents répond à une proposition ou contre-proposition. Quatre cas sont alors à considérer :

- *Cas1* : l'offre du partenaire est acceptée et conséquemment, la négociation se termine.

Il n'y a pas de concession à faire, alors $\omega(t_i) = 0$

$\Omega(t_i) = \Sigma^i_{j=1}\ \omega(t_j) = \Sigma^{i-1}_{j=1}\ \omega(t_j) + \omega(t_i) = \Sigma^{i-1}_{j=1}\ \omega(t_j) = \Omega(t_{i-1})$

$\Rightarrow \Omega(t_i) = \Omega(t_{i-1})$

$\Rightarrow$ pas de nouvelle offre faite par l'agent

- *Cas2* : l'offre du partenaire n'est pas acceptée et l'agent décide de mettre fin à la négociation.

Il n'y a pas de concession à faire $\omega(t_i) = 0$

$\Rightarrow \Omega(t_i) = \Omega(t_{i-1})$

$\Rightarrow$ pas de nouvelle offre faite par l'agent

- *Cas3* : l'offre du partenaire n'est pas acceptée et l'agent décide de répéter la même offre qu'à l'étape précédente.

Il n'y a pas de concession à faire $\omega(t_i) = 0$

$\Rightarrow \Omega(t_i) = \Omega(t_{i-1})$

$\Rightarrow \delta(t_i) = \delta(t_{i-1})$

- *Cas4* : l'offre du partenaire n'est pas acceptée, mais l'agent décide de faire une concession $\omega(t_i) > 0$.

Alors,

$\Omega(t_i) = \Sigma^i_{j=1}\ \omega(t_j) = \Sigma^{i-1}_{j=1}\ \omega(t_{j-1}) + \omega(t_i) = \Omega(t_{i-1}) + \omega(t_i)$

$\Rightarrow \Omega(t_i) > \Omega(t_{i-1})$, et

la nouvelle offre $\delta(t_i) = \delta(t_{i-1}) + \omega(t_i)$

$\Rightarrow \delta(t_i) > \delta(t_{i-1})$

Conclusion 1 *: dans tous les cas, la concession globale pour chaque agent est une série monotone croissante. De même, les offres faites par les agents suivent une série monotone croissante et les utilités individuelles des agents suivent une série monotone décroissante.*

Théorème 1 *: si les agents négociateurs utilisent tous deux la stratégie de négociation proposée ci-dessus, alors s'ils concluent un contrat sur une offre δ, ce contrat maximise le produit de leurs utilités respectives telles qu'estimées par chacun des agents.*

Si on considère le produit des utilités des agents telles que vues par chacun des agents (A_i, A_j) au cours de la négociation, on voit facilement que ce produit constitue une série monotone croissante. En effet, à chaque étape de la négociation (avant d'atteindre l'étape finale) seul un des agents fait une concession. Sans perdre de généralité[39], on peut supposer que c'est l'agent A_i qui fait la concession minimale suffisante.

Si on dénote par $Risque^i_i$ et U^i_i, respectivement le risque et l'utilité de l'agent A_i tel qu'estimé par lui-même, et $Risque^j_i$ et U^j_i, respectivement le risque et l'utilité de A_j tel qu'estimé par A_i.

<u>À l'étape t</u> :

c'est l'agent A_i qui fait la concession, on a donc $Risque^i_i(t) < Risque^j_i(t)$

$\Rightarrow [U^i_i(\delta_i(t)) - U^i_i(\delta_j(t))] / [U^i_i(\delta_i(t))] < [U^j_i(\delta_j(t)) - U^j_i(\delta_i(t))] / [U^j_i(\delta_j(t))]$

$\Rightarrow [U^i_i(\delta_i(t)) * U^j_i(\delta_j(t))] - [U^i_i(\delta_j(t)) * U^j_i(\delta_j(t))] < [U^i_i(\delta_i(t)) * U^j_i(\delta_j(t))] - [U^i_i(\delta_i(t)) * U^j_i(\delta_i(t))]$

$\Rightarrow U^i_i(\delta_j(t)) * U^j_i(\delta_j(t)) > U^i_i(\delta_i(t)) * U^j_i(\delta_i(t))$

$\Rightarrow \Pi(\delta_j(t)) > \Pi(\delta_i(t))$ (I)

où Π est le produit des utilités des agents A_i et A_j telles qu'estimées par A_i.

<u>À l'étape $(t+1)$</u> :

l'agent A_i aurait déjà fait la concession minimale suffisante pour balancer le risque de son partenaire A_j à l'étape t, on aura donc : $Risque^i_i(t+1) > Risque^j_i(t+1)$.

---

[39] Le même raisonnement peut s'appliquer aussi pour Aj.

$\Rightarrow [U^i_i(\delta_i(t+1)) - U^i_i(\delta_j(t+1))] / [U^i_i(\delta_i(t+1))] > [U^j_i(\delta_j(t+1)) - U^j_i(\delta_i(t+1))] / [U^j_i(\delta_j(t+1))]$

$\Rightarrow [U^i_i(\delta_i(t+1)) * U^j_i(\delta_j(t+1))] - [U^i_i(\delta_j(t+1)) * U^j_i(\delta_j(t+1))] > [U^i_i(\delta_i(t+1)) * U^j_i(\delta_j(t+1))] - [U^i_i(\delta_i(t+1)) * U^j_i(\delta_i(t+1))]$

$\Rightarrow U^i_i(\delta_i(t+1)) * U^j_i(\delta_i(t+1)) > U^i_i(\delta_j(t+1)) * U^j_i(\delta_j(t+1))$

$\Rightarrow \Pi(\delta_i(t+1)) > \Pi(\delta_j(t+1))$ *(II)*

En analysant les étapes *t* et *(t+1)* on peut facilement voir que : d'une part, à l'étape *t* c'est l'agent *Ai* qui fait la concession, c'est à dire que si la concession n'est pas faite, c'est l'offre de *Aj*, $\delta_j(t)$ qui sera retenue et en même temps, c'est elle qui maximise le produit des utilités des agents, *équation (I)*. D'autre part, à l'étape *(t+1)* c'est *Aj* qui doit faire une concession, alors si la concession n'est pas faite, c'est l'offre de *Ai*, $\delta(t+1)$ qui sera retenue et en même temps c'est elle qui maximise le produit des utilités des agents, *équation (II)*.

***Conclusion 2** : d'après les équations (I) et (II), on a le maximum des produits des utilités des agents : { Max [$\Pi(\delta_i(t))$, $\Pi(\delta_j(t))$], t=1, 2, ...} est une série monotone croissante.*

Pour démontrer le *théorème 1*, supposons que les deux agents concluent une entente à l'étape *t* sur une offre δ. Notons $\Pi(\delta)$ le produit des utilités des agents relativement à l'offre δ.

Si on suppose que δ n'est pas l'offre acceptable qui maximise le produit des utilités des agents *Ai* et *Aj*, alors, il doit exister une offre δ' acceptable pour les deux agents tel que $\Pi(\delta') > \Pi(\delta)$. Cependant, à cause de la minimalité de la concession que font les agents, δ' devrait déjà avoir été utilisée par l'un des agents dans une étape précédente de la négociation.

Sans perdre de généralité supposons que δ est une offre de *Ai*, puisque $\Pi(\delta') > \Pi(\delta)$ et δ' n'a pas été utilisé par *Ai* donc elle devrait permettre à *Aj* de réaliser une meilleure utilité que celle qu'il va recevoir en acceptant δ. C'est à dire qu'avec δ', *Aj* aurait à faire moins de concession. Mais puisque à chaque étape, chaque agent choisit l'offre qui lui permet de faire la concession minimale suffisante et, comme à l'étape t les agents ont

conclu une entente sur δ, alors δ' devrait être déjà utilisé par *Aj* dans une étape précédente avant d'arriver à l'offre δ. Cela contredit la *conclusion 2* de même que le *lemme 1* qui affirment que les δ et les $\Pi(\delta)$ suivent une série monotone croissante.

Note *:* dans le modèle que nous avons utiliser pour l'implantation de la stratégie de négociation, nous approchons seulement l'offre qui maximise le produit des utilités des agents. En effet, le degré d'approximation dans les offres dépend du degré de précision choisit par le développeur pour l'évaluation des risques.

Conclusion3 *: la stratégie de négociation basée sur la concession monotone maximise le produit des utilités des agents mais ne maximise pas la somme de leurs utilités, en fait, il y a problème sur l'efficacité globale.*

Pour illustrer le problème de maximisation du profit conjoint, considérons l'exemple suivant : soit deux agents *Ai* (acheteur) et *Aj* (vendeur) qui négocient la qualité d'un service de communication. Le tableau suivant résume l'utilité que peut avoir chacun des agents selon la qualité du service qui peut varier de 1 à 5.

Tableau 5.1: scénario de négociation de la qualité d'un service

Qualité	Utilité de l'agent Ai	Utilité de l'agent Aj
1	0	10
2	1	3
3	2	2
4	3	1
5	10	0

Si les deux agents utilisent la stratégie de négociation présentée, ils vont conclure un contrat sur la qualité 3 qui procure à chacun une utilité de 2. Ce contrat maximise le produit de leurs utilités (4 au lieu de 3 ou 0), mais ne maximise pas la somme de leurs utilités (4 au lieu de 10) qu'ils auraient pu accomplir en utilisant le premier ou le dernier choix du tableau 5.1.

Nous allons revenir sur la maximisation du bien être du système (appelé aussi profit conjoint des agents) lors de l'évaluation des essais expérimentaux (cf. 7).

b- la stabilité de la stratégie : une stratégie *S* est dite stable [Fudenberg 1991] si la condition suivante est vérifiée : « si un agent *Ai* utilise la stratégie *S*, son partenaire *Aj* ne peut pas faire plus de profit en utilisant une stratégie différente de *S* ».

Ceci est particulièrement important pour le développeur d'agents intelligents et du commerce électronique. En effet, sous cette condition, il n'y a plus nécessité de cacher sa stratégie. La stratégie peut être publiquement connue sans qu'aucun ne puisse exploiter l'information pour choisir une stratégie différente. Il est même préférable que la stratégie de négociation soit connue pour garder une homogénéité dans l'interaction des agents.

Pour la stratégie décrite ci-dessus, seul un des agents doit faire une concession à chaque étape dépendamment de son *Risque* et cette concession doit être minimale et suffisante. Ainsi, à chaque étape le choix de concéder ou non ne dépend que de l'attitude de l'agent à risquer une situation de conflit. Si les risques sont différents, l'agent qui doit concéder est toujours connu, donc pas de problème de stabilité. Mais le problème se pose quand les risques des deux agents sont équivalents, puisqu'il faut alors un mécanisme pour déterminer qui doit concéder. En fait, si on considère une situation où les deux agents se trouvent avec des *Risques* équivalents et à une seule étape de conclure un contrat, il suffit que l'un d'eux fasse une concession pour que les deux agents arrivent à une entente. Cependant, si l'un d'eux sait que son partenaire utilise une telle stratégie, il peut ne pas faire de concession et laisser son partenaire faire la concession nécessaire pour arriver à l'entente. Ainsi, la stratégie décrite ci-dessus n'est pas stable pour ce cas particulier.

Dans notre implantation, afin d'assurer la stabilité de la stratégie, on a convenu que seul l'agent acheteur peut répéter une même offre sans faire de concession[40]. L'agent vendeur doit toujours faire une concession, tant que c'est possible[41]. En fait, la stabilité a été assurée au dépend du profit individuel du vendeur.

c- simplicité de la stratégie : le mécanisme de négociation basé sur la présente stratégie n'est pas particulièrement simple. En fait cela est du à deux facteurs principaux.

---

[40] L'agent acheteur ne peut pas terminer une négociation en envoyant un message « reject » au vendeur, mais il peut répéter la même offre s'il ne veut plus faire de concession.

[41] L'agent vendeur ne peut pas répéter une même offre, mais il peut arrêter la négociation s'il ne peut plus faire de concession.

Premièrement, les agents doivent effectuer tous les calculs pour évaluer les fonctions d'utilité et les risques, et dépendamment du degré de précision cela peut nécessiter un très grand nombre d'interactions.

Deuxièmement, le processus de négociation nécessite plusieurs étapes avant de converger à une entente, ce qui nécessite l'échange d'un très grand nombre de messages entre les agents.

En fait, la charge computationnelle et la charge de communication constituent un sérieux désavantage pour une telle stratégie (les résultats expérimentaux le confirment : cf. 7.2).

Dans notre implantation, le degré de précision des solutions et le nombre d'itérations peuvent être ajusté selon le choix de l'utilisateur. Ainsi, l'utilisateur peut décider de l'incertitude acceptable pour l'évaluation des risques et du nombre d'itérations maximales après lesquelles l'agent propose directement son offre maximale. Cela accélère la convergence du processus de négociation.

Chapitre 6: MIAMAP[42] : une place commerciale virtuelle pour agents mobiles intelligents

6.1 Introduction

Ces dernières années, plusieurs chercheurs dans le domaine des agents intelligents se sont intéressés à la conception de marchés virtuels pour le commerce électronique (E-commerce) [Maes 1996, Rodriguez 1997, Tsvetovatyy 1997], et à l'étude des protocoles régissant l'interaction des agents impliqués dans de telles transactions. En analysant la littérature relative aux marchés virtuels, nous avons noté plusieurs exigences nécessaires au développement de tels systèmes :

- fournir le support nécessaire pour une large variété de transactions, allant d'une simple opération de vente / achat à une négociation complexe impliquant plusieurs agents,
- fournir un langage de communication suffisamment riche pour permettre d'exprimer la sémantique reliée au commerce,
- être extensible par l'intégration de partie tiers. Ainsi, la médiation dynamique ou la construction de contrats multiagent sera possible,
- fournir un mécanisme de paiement sécuritaire et fiable,
- contrôler les fraudes et les fausses représentations.

Cependant, même si les architectures existantes fournissent les mécanismes pour une interaction directe entre les agents, elles restent déficientes d'une facilité explicite pour supporter les protocoles de négociations. En effet, ces architectures ne fournissent pas de protocoles comme partie intégrante de la plate-forme.

[42] MIAMAP : pour designer MIAMI Market Place.

Dans le cadre du projet MIAMI (cf. 1.2) nous avons conçu et implanté une place commerciale virtuelle (MIAMAP) [Esmahi 1999b, Esmahi 2000a] qui intègre de tels protocoles.

Dans ce chapitre, nous présentons les concepts et mécanismes utilisés pour l'implantation de MIAMAP et des agents courtiers représentant les différents domaines de la plate-forme MIAMI.

6.2 Architecture de la place commerciale

Dans la plate-forme MIAMI, la place commerciale constitue l'infrastructure principale pour l'interaction des agents (échange d'offres et demandes). Les agents représentant les différents acteurs dans MIAMI, peuvent alors annoncer leurs services disponibles, solliciter des offres auprès des fournisseurs ou faire des offres aux clients potentiels. Trois types d'agents sont donc en interaction dans la place commerciale : le gestionnaire de la place ou d'un domaine d'activité dans la place, les agents acheteurs et les agents vendeurs. Tout au long de ce chapitre, nous employons les termes *acheteur* et *vendeur* pour distinguer les deux différents rôles que peut jouer un agent client dans la place. Toutefois, un agent peut être dans l'un des états suivants : (1) acheteur et vendeur en même temps (i.e. agent AVP ou VE), (2) acheteur seulement (i.e. Utilisateur de service) ou (3) vendeur seulement (i.e. Agent CP).

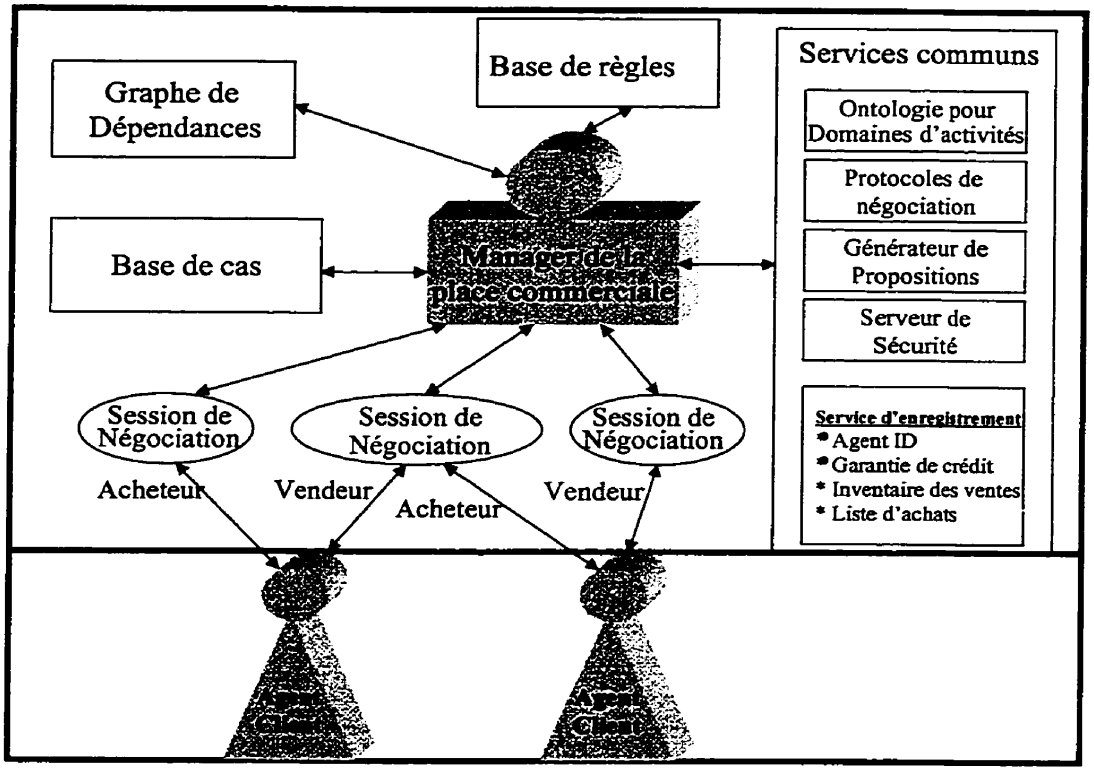


Figure 6.1: Architecture de MIAMAP

L'agent manager, agit comme facilitateur en administrant les services de la place commerciale et comme médiateur dans les conflits entre acheteurs et vendeurs. Ainsi, tous les agents doivent s'enregistrer auprès du manager avant qu'ils ne puissent effectuer des interactions dans la place commerciale. La place commerciale fournit un ensemble de services communs pour faciliter et sécuriser les opérations commerciales parmi lesquels on peut citer :

- un service d'enregistrement des agents clients de la place. Pour son enregistrement, chaque agent doit fournir ou se voir attribuer une identité ainsi que l'inventaire des services offerts ou demandés. Si l'agent est un acheteur, il doit fournir une garantie de crédit qui fixera la limite supérieure de paiement qu'il peut effectuer,
- une spécification des protocoles[43] de négociation supportés par la facilité de négociation,

[43] Contract-net, FIPA-iterated contract-net, English auction, Dutch auction, Yankee auction.

- une ontologie [Gruber 1992, Gruber 1993] pour chaque domaine d'activité existant dans la place. En fait, chaque domaine correspond à une communauté d'agents qui adhèrent à cette activité. Ces derniers utilisent une telle ontologie pour leurs communications,
- un générateur de proposition, qui peut être utilisé par les agents pour construire leurs propositions initiales ou par le manager dans son processus de médiation,
- un service de sécurité pour la protection contre la fraude et la fausse représentation.

L'activité au sein de la place commerciale est organisée autour de trois concepts: les services fournis par la place, les agents (manager et clients) et les sessions d'interactions. Ainsi, pour être reconnu comme élément actif dans la place, chaque agent lance son processus principal qui reste en écoute en se réveillant à des intervalles de temps réguliers. La première session de négociation prend place quand un des agents acheteurs s'active, sélectionne un élément de sa liste d'achat et demande au manager de lui recommander un agent fournisseur. Si plus d'un agent a annoncé sa capacité de vendre l'article demandé, le manager utilise des critères de sélection pour choisir la meilleure offre qui correspond à la demande (i.e. correspondance sur les caractéristiques de l'article, prix, dépendances entre les agents).

La communication entre les agents dans la place commerciale utilise les langages FIPA-ACL/KQML et les mécanismes de traitement des événements « *Event-listener* » fournis par le langage JAVA [Bigus 1997, Orfali 1998]. Nous avons choisi d'implanter une combinaison de FIPA-ACL [FIPA 1997a, FIPA 1998] et de KQML [Labrou 1996, Labrou 1997] pour les raisons suivantes :

- nous avons besoin en même temps d'un service de courtage et d'un service de négociation. Cependant, aucun des deux langages n'offre des performatives pour les deux services en même temps,
- KQML fournit toutes les performatives dont nous avons besoin pour le courtage comme : *Recommend-One, Register, Advertise, Ask, Reply,*
- FIPA-ACL fournit les performatives pour la négociation comme : *CFP (Call For Proposal), Propose, Reject-Proposal, Accept-Propopsal, Confirm,*

- FIPA-ACL définit en utilisant ces performatives un ensemble de protocoles (Contract-net, FIPA-iterated contract-net, English auction, Dutch auction) ce qui nous a facilité l'implantation de notre composante protocole de négociation décrite dans l'architecture de MIAMAP,
- notre objectif n'est pas d'étudier le langage de communication mais plutôt de proposer un protocole de négociation.

Ainsi, chaque événement de communication adressé d'un agent à un autre comprend un message FIPA-ACL/KQML. La figure 6.2 illustre les différents messages échangés entre les différents agents acteurs dans la place commerciale.

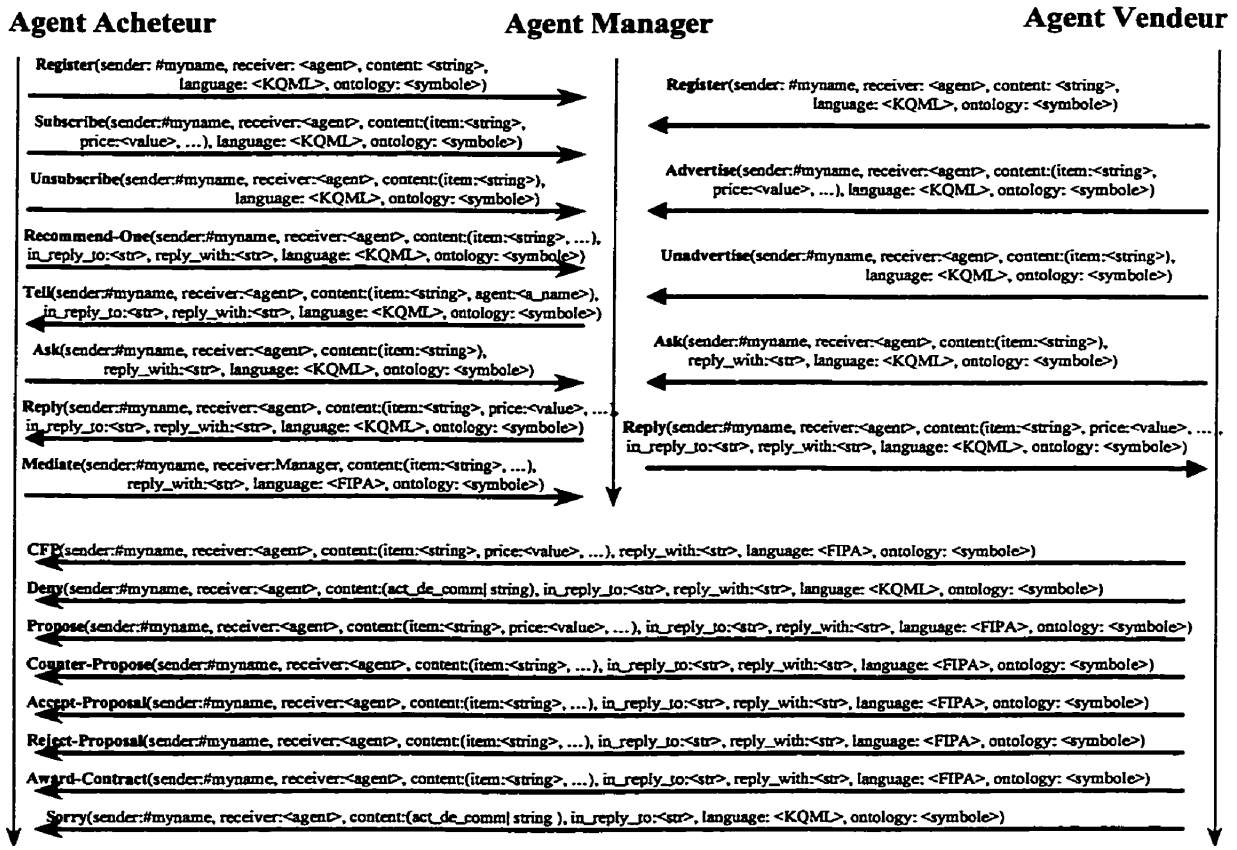


Figure 6.2: Diagramme de séquence représentant l'interaction dans MIAMAP.

6.3 Architecture d'implantation de l'agent manager

L'agent manager agit en tant que superviseur de la place commerciale en assurant le contrôle des différents services et en agissant comme facilitateur des différentes sessions de négociation en cours.

Afin de gérer le service d'enregistrement, le manager maintient deux tables (*Hashtables*), une qui sert de registres pour tous les agents actifs dans le marché et l'autre pour les communautés d'agents. Ces communautés correspondent aux différents domaines d'activité supportés par la place. Pendant leur enregistrement, les agents fournisseurs sont ajoutés à ces communautés en fonction de l'annonce qu'ils font sur leur capacité de fournir un service dans chacun des domaines de leur intérêt. Ainsi, un agent doit toujours retirer son adhésion à une communauté pour laquelle il n'a plus d'intérêt. Une autre alternative paraît plus pratique, que nous n'avons pas encore implantée, consiste à permettre au manager de retirer l'adhésion d'un agent qui répond toujours négativement aux annonces adressées à la communauté.

Dans cette première version de la place commerciale, nous n'avons implanté qu'un seul agent manager pour tous les domaines. Nous utilisons la propriété de singleton dans son implantation, ce qui assure qu'une seule instance de celui-ci sera créée dans la place commerciale. Ceci nous a aussi facilité l'implantation du contrôle des différents services du marché puisque le manager est exécuté dans le processus principal de la place commerciale et les événements générés par les agents clients sont traités de façon synchrone. Une conception alternative pour la gestion de la place commerciale, consiste à implanter un manager pour chaque domaine d'activité et de déclarer un nouveau *thread* à chaque fois qu'un événement doit être traité pour éviter de créer un point de congestion au niveau du manager. En fait, nous avons choisi de simplifier notre implantation puisque notre principal but est d'abord de tester le modèle de négociation que nous avons proposé, et en même temps ce premier prototype va nous permettre de prendre des mesures sur la charge d'activité du manager pour valider le choix de ces autres solutions alternatives.

La figure 6.3 présente l'architecture d'implantation de l'agent manager.

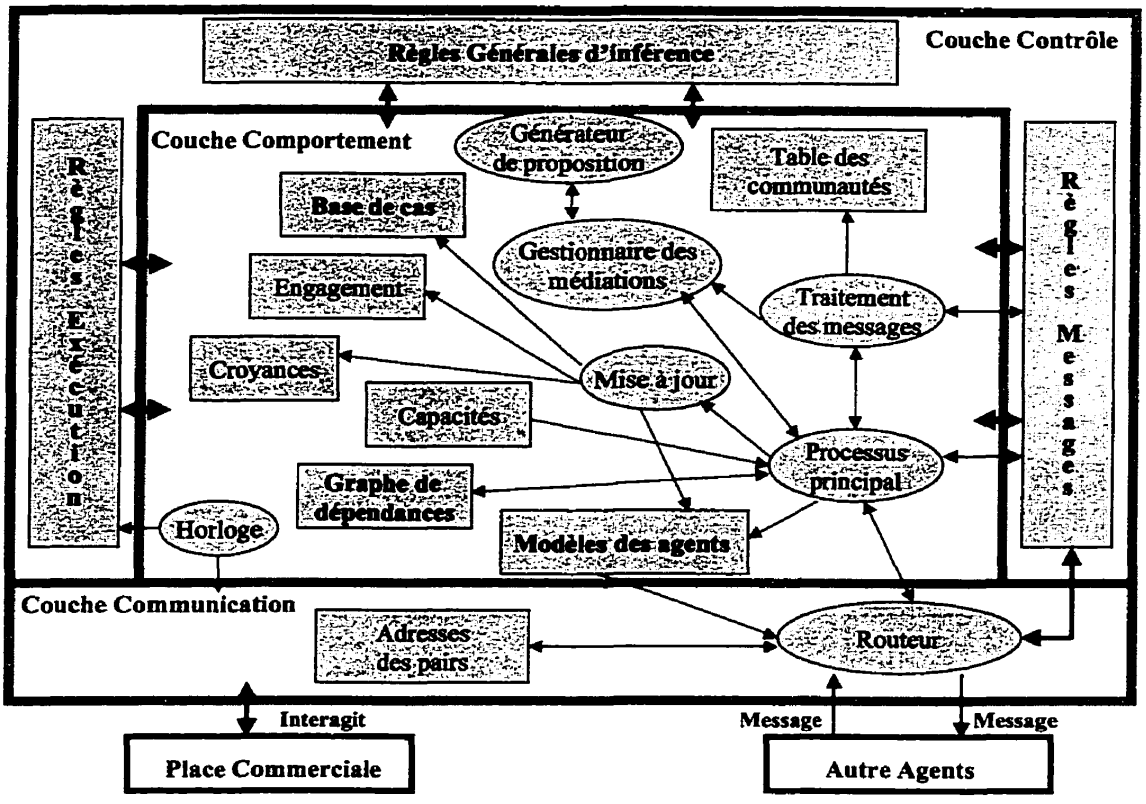


Figure 6.3: Architecture de l'agent manager

La couche communication : la communication entre les agents est accomplie en utilisant un mécanisme d'écoute d'événement [Bigus 1997]. Chaque événement contient deux arguments : l'identité de l'agent qui a déclenché l'événement et un objet correspondant au message FIPA-ACL/KQML envoyé. Une fois le message extrait de l'événement, le processus de routage est déclenché pour le traiter. Le routeur de la couche communication possède un comportement par défaut face à six types de performatives : *'register'*, *'advertise'*, *'unadvertise'*, *'subscribe'*, *'unsubscribe'*, *'sorry'*. Les autres types de performatives sont traiter au niveau de la couche de contrôle. Le routeur du manager fournit aussi aux autres agents un service de diffusion et d'acheminement de messages entre eux. Toutefois, les agents peuvent communiquer aussi de façon directe.

La performative *'register'* permet d'enregistrer l'agent expéditeur comme étant un agent actif dans la place commerciale. Les performatives *'advertise'* et *'unadvertise'*

permettent respectivement d'enregistrer l'agent auprès d'une communauté (domaine d'activité) ou de le retirer de celle-ci. Les performatives *'subscribe'* et *'unsubscribe'* permettent d'ajouter ou de supprimer des articles à la liste d'achat de l'agent. La performative *'sorry'* ne déclenche aucun traitement spécial, mais il affecte une valeur *'null'* au paramètre qui doit recevoir le résultat de la requête qui a causé le *'sorry'*.

Les messages de communication sont constitués d'une collection de données correspondant aux principaux attributs spécifiés dans FIPA-ACL/KQML. Le premier élément du message appelé performative est un verbe identifiant l'acte de discours qui définit la signification du message. La performative est suivie par une séquence de paramètres présentés par des mots clés, figure 6.2. L'un des paramètres décrit le contenu du message par une expression encodée dans un formalisme défini. Les autres paramètres sont soit : des attributs qui aident le service de transport à délivrer le message correctement en identifiant la source *'sender'* et la destination *'receiver'*, des attributs qui aident l'agent récepteur à interpréter correctement la signification du message, par exemple : *'language'* et *'ontology'* ou des paramètres qui aident les agents à interagir de façon coopérative, par exemple *'reply-with'* et *'in-replyto'*.

Couche comportement : cette couche implante les fonctions de comportement du manager. Les procédures correspondantes sont déclenchées pour le traitement de messages reçus ou pour l'exécution de tâches planifiées à des instants donnés. Cette couche comporte une horloge temps réel utilisée pour l'exécution des actions planifiées (actions de communications ou actions locales) ou pour la mise à jour de l'état mental du manager. L'état mental du manager est composé de croyances, engagements et capacités. L'ensemble des croyances constitue la base de connaissance de l'agent, c'est aussi le modèle que possède le manager de son environnement (place commerciale). Chaque croyance correspond à un fait temporel que le manager considère être vrai ou faux à un moment donné. Les engagements représentent l'ensemble des actions que le manager avait planifié d'exécuter, soit pour le profit d'un autre agent soit pour ses opérations de gestion de la place commerciale. Les capacités représentent l'ensemble des actions que le manager est en mesure d'accomplir, soit pour fournir un service aux agents clients de la

place commerciale ou pour effectuer ses fonctions de manager. Chaque action est représentée par son nom d'appel et la liste de ses arguments. Les actions sont généralement appelées par des règles de type détecteur (*Sensor*) ou actuateur (*Effector*) (couche contrôle). La composante modèles des agents, contient les connaissances que le manager représente au sujet de chaque agent enregistré dans la place commerciale (i.e. nom de l'agent, inventaire de ses ventes, liste de ses achats, domaine d'activité, crédit de paiement). Les connaissances sur les actions que le manager peut effectuer au profit des agents clients doivent être intégrées par les développeurs de ces derniers sous formes de messages FIPA/KQML que les agents peuvent adresser au manager. En fait, cet ensemble de messages constitue le modèle que les agents clients possèdent du manager.

Le comportement du manager face à ces messages dépend de son état mental et du type de message reçu. Ainsi, un message *'register'* entraîne la mise à jour des registres du marché. Si le message est de type *'subscribe', 'usubscribe', 'advertise'* ou *'unadvertise'*, le manager met à jour le modèle de l'agent, le registre des communautés et peut être aussi le graphe de dépendances des agents. Si le message est une requête d'information (*'recommend-one'* ou *'ask'*), l'agent essaie d'unifier le contenu du message à ses croyances et prend la décision de retourner le résultat au demandeur. Si le message est un *'mediate'*, le manager prend la décision de démarrer une session de médiation entre le fournisseur et l'acheteur. Finalement, si le message est une demande de service (*'achieve'*) et l'action spécifiée dans le message fait partie des capacités du manager, il décide de prendre l'engagement pour accomplir l'action. Tous les messages qui nécessitent une prise de décision de la part du manager sont sous le contrôle des règles de la couche de contrôle.

La composante graphe de dépendances représente toutes les transactions possibles entre les agents qui se sont enregistrés comme acheteurs dans la place commerciale. Une telle structure est représentée par un graphe orienté où chaque nœud correspond à un agent et les arcs joignant les nœuds correspondant aux dépendances entre les agents en terme de transactions potentielles sur des articles. À chaque arc sont associés l'identifiant de l'article objet de la transaction et le prix demandé par l'agent fournisseur. Les arcs sont

orientés et partent du nœud représentant l'agent demandeur vers le nœud représentant l'agent fournisseur.

La figure suivante décrit un scénario de la place commerciale et son graphe de dépendances correspondant.

Tableau 6.1: un scénario d'interaction dans la place commerciale

Articles	Agent A1		Agent A2		Agent A3		Agent A4	
	Prix de Vente	Prix D'achat	Prix de Vente	Prix D'achat	Prix de Vente	Prix D'achat	Prix de Vente	Prix D'achat
I1		X	P1					X
I2		X	P2					
I3	P3			X	P6			
I4		X					P4	
I5		X			P5			
I6				X			P7	
I7	P8					X		
I8					P9			X
I9					P10			X
I8+I9					P11			X

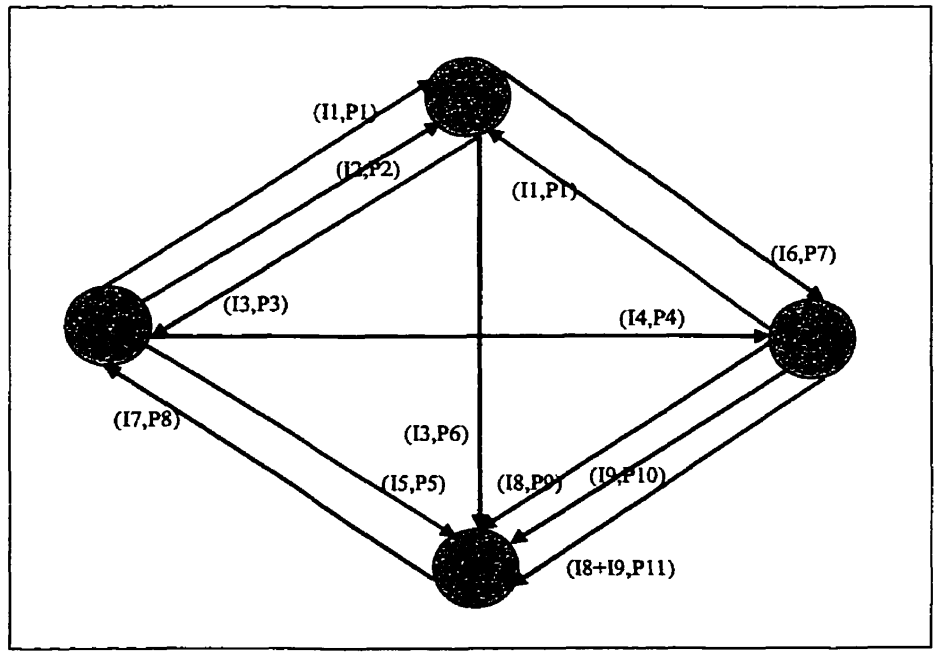


Figure 6.4: graphe de dépendances correspondant au scénario du tableau 6.1

Le manager maintient aussi pour son raisonnement interne une base de cas que nous allons décrire brièvement, même si son implantation n'est pas encore faite pour ce premier prototype de la place commerciale. Cette base vient compléter les connaissances de l'agent déjà intégrées sous formes de croyances, en lui fournissant une base de cas de négociations qu'il utilise dans toutes les activités nécessitent un raisonnement par cas (génération de solution initiale, médiation). Cette base de cas comprend aussi bien les cas de négociations réussis (contrat) que les cas de négociations échoués. L'historique de chaque cas est représenté par un vecteur d'états, chaque état est un objet qui représente la proposition d'un des agents négociateurs, le feed-back des autres (acceptation ou rejet), la cause du rejet si elle existe, et la modification apportée pour améliorer la proposition ou les arguments utilisés pour convaincre les agents partenaires à l'accepter. Les cas de succès sont indexés selon les caractéristiques pertinentes du domaine (i.e. : nom produit / service, caractéristiques du produit, prix, date, agent vendeur, agent acheteur), les cas d'échecs possèdent deux autres indexes supplémentaires : le type d'échec et la cause qui a entraîné l'échec. Les cas sont organisés en hiérarchie autour des concepts importants du domaine (mots clés), et leur recherche se fait selon les similarités conceptuelles. Le niveau le plus haut de la hiérarchie est appelé '*épisode généralisé*' [Kolodner 93] et chaque nœud est constitué soit d'un cas individuel soit d'un épisode généralisé.

Couche contrôle : cette couche a pour rôle de faciliter la spécification du comportement du manager. En effet, elle permet d'implanter une architecture délibérative où l'état mental et le comportement du manager sont contrôlés par des règles. La base de règles encodée dans cette couche est composée de trois ensemble de règles : les règles de traitement des messages, les règles d'exécution des tâches et les règles générales d'inférence. Ce dernier ensemble de règles permet au manager de faire des déductions d'ordre général basées sur son état mental. Le comportement délibératif du manager dépend de sa base de règles et de son état mental. Les messages reçus dans le routeur sont comparés aux règles de traitement des messages. Si l'une des règles est satisfaite, elle sera sélectionnée et exécutée, sinon le comportement par défaut est appliqué. Lorsqu'une action est sur le point d'être exécutée, l'ensemble de règles d'exécution des tâches est

consulté pour déterminer la procédure à lancer. Si aucune règle n'est satisfaite, l'exécution de l'action échouera. Finalement, lorsque aucun message n'est en attente et qu'il n'y a pas d'action à exécuter, le manager déclenchera les règles générales d'inférence jusqu'à ce que l'une des trois conditions suivantes soit satisfaite : (1) un nouveau message est arrivé, (2) une action doit être exécutée ou (3) aucune règle générale n'est satisfaite. Enfin, si aucune règle d'inférence ne peut être déclenchée et qu'aucun autre événement n'est prévu, le manager calcule le temps prévisible du prochain événement et se met en position d'attente passive (s'endort).

L'annexe II décrit un exemple concret de la définition de l'état mental de l'agent et de l'utilisation des règles pour contrôler son comportement.

6.3.1 Classes d'objets utilisés pour l'implantation du manager

Afin d'implanter aussi bien l'agent manager que les autres agents que nous avons utilisés dans l'évaluation du modèle de négociation de MIAMAP, nous avons définit un ensemble de classes objets. Parmi ces classes, une classe abstraite *'BasicAgent'* qui est héritée par tous les agents. La classe *'BasicAgent'* implante les fonctionnalités de la couche communication et le comportement réactif des agents.

Pour implanter le comportement délibératif et l'état mental de l'agent, nous avons défini une classe base de connaissance '*KnowledgeBase*' et nous avons déclaré trois groupes de règles que nous appelons : *BeliefsKB, CapabilitiesKB et CommitmentsKB*. Le groupe '*BeliefsKB*' définit les règles générales d'inférence et les croyances de l'agent, le groupe '*CommitmentsKB*' définit les règles de gestion des messages et les engagements prises suite au traitement de ces messages. Le groupe '*CapabilitiesKB*' définit les règles d'exécution des tâches pour accomplir les engagements.

Pour déclarer les éléments de la base de connaissance nous avons défini une classe pour les règles '*Rule*', une classe pour les faits '*Fact*', une classe pour les clauses '*Clause'* et deux interfaces pour définir des actuateurs '*Effector'* et des détecteurs '*Sensor'*.

Les règles de traitement des messages utilisent dans leur partie condition un détecteur pour appeler un processus de traitement de message et dans leur partie conséquence un actuateur qui peut soit déclencher une action (i.e. : l'envoi d'une réponse) soit procéder à

la création d'un engagement sous forme d'une nouvelle croyance (fait). Une règle de traitement de message ne doit comporter qu'une seule condition de réception de message mais celle-ci peut être combinée avec des conditions sur les croyances ou les engagements.

Les règles d'exécution des tâches utilisent aussi un détecteur dans leur partie condition pour vérifier que les conditions de l'état mental de l'agent lui permettent d'exécuter les tâches relatives à un engagement, et un actuateur pour lancer ces tâches. Un seul engagement peut être spécifié dans la règle.

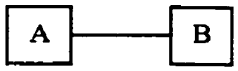
Les règles générales d'inférence n'utilisent ni détecteur ni actuateur et sont appliquées seulement sur les faits représentant les croyances de l'agent et non pas sur les faits relatifs aux engagements.

Dans la classe définissant la base de connaissance '*KnowledgeBase*', nous avons intégré les procédures définissant un moteur d'inférence utilisant le chaînage avant et chaînage arrière.

La hiérarchie de classes utilisée pour l'implantation du manager est donnée par la figure 6.5 et le diagramme d'agrégation de ces classes est donné par la figure 6.6.

Légende : Dans la suite de ce chapitre, nous allons utiliser la notation UML [OMG 1999] dans nos diagrammes de classes et les descriptifs suivants pour définir les propriétés des relations entre les classes :

Relations :



: association entre les classes A et B

: B est une composante de A

: B est une sous-classe de A

Cardinalité :

1 : un et un seul

0..1 : zéro ou un

M..N : de M à N (entiers naturels)

* : plusieurs

0..* : de zéro à plusieurs

1..* : de un à plusieurs

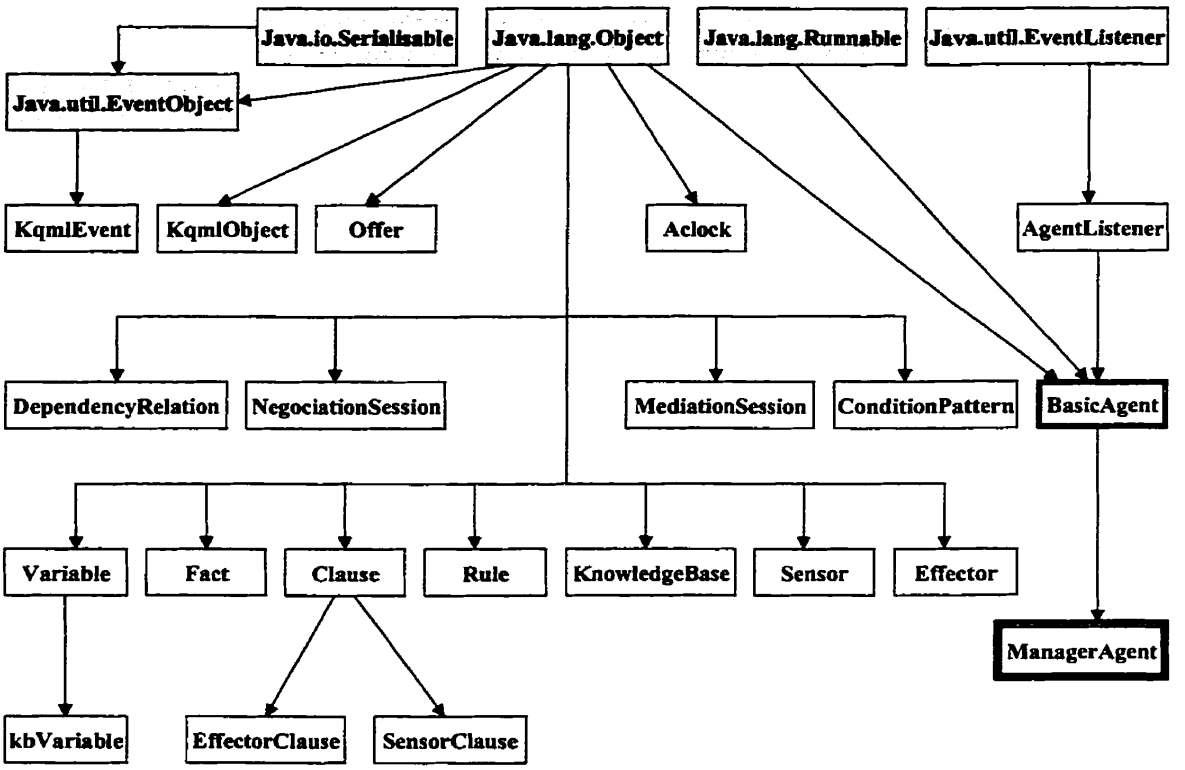


Figure 6.5: Hiérarchie d'héritage pour les classes objets du manager

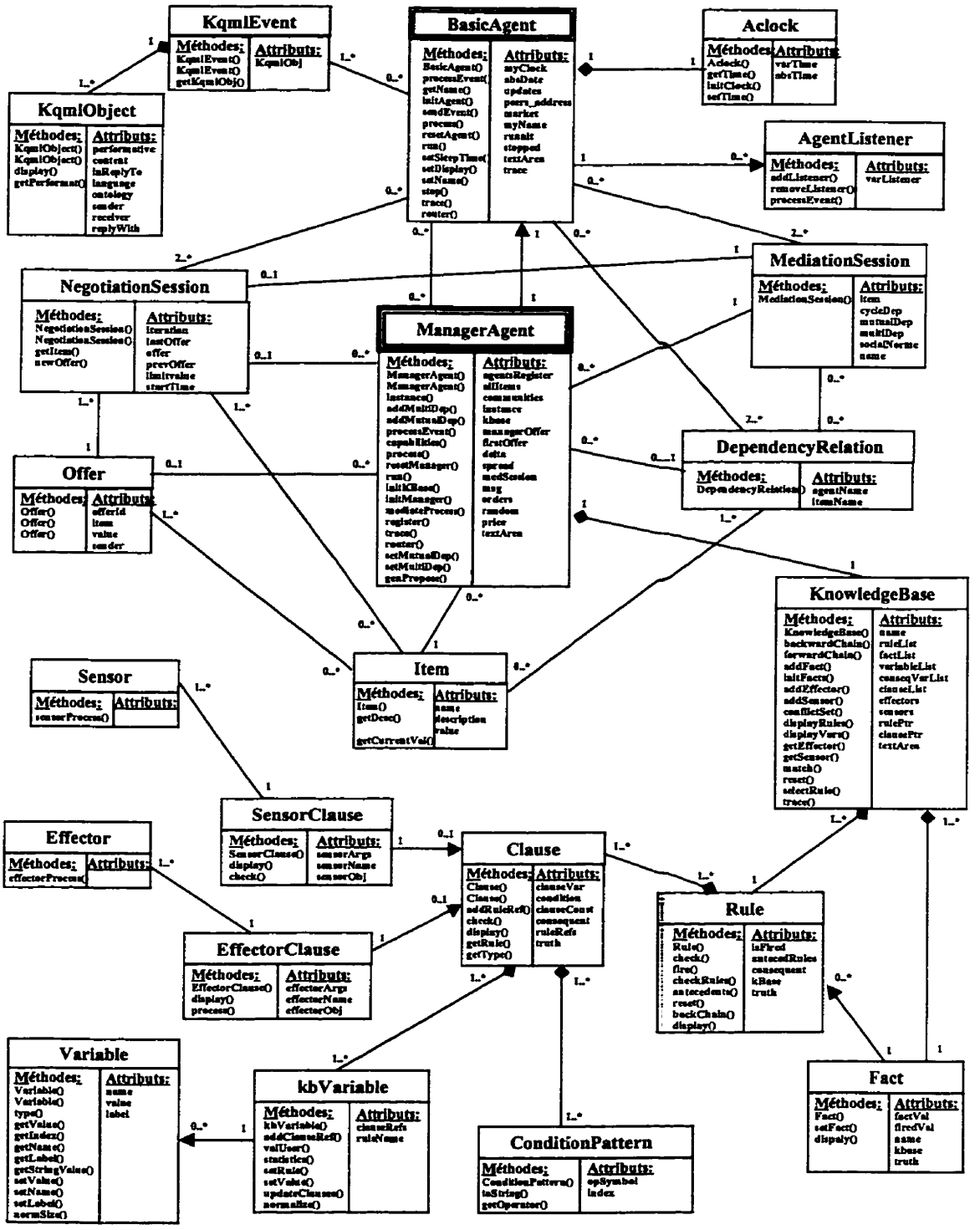


Figure 6.6: Diagramme d'agrégation des classe objets du manager

Classe BasicAgent : c'est une classe abstraite qui définit une interface d'implantation commune et implante le comportement de base pour tous les agents de notre système. L'architecture de classe *'BasicAgent'* est fondée sur quatre éléments fondamentaux : le processus principal de l'agent, le routeur de communication, l'adresse des pairs et l'horloge interne de l'agent. Le routeur est le processus chargé de la réception et de l'expédition de messages. L'adresse des pairs est un vecteur contenant des références aux autres agents. L'horloge est un objet de la classe *'Aclock'* et fournit le temps. Elle est liée à l'horloge du système. Finalement, le processus de l'agent définit le *thread* principale de l'agent. En fait, la classe *'BasicAgent'* implémente l'interface *'Runnable'* de JAVA, ce qui nécessite des sous-classes de cette classe, de définir une méthode *'Run()'* qui implante le corps du *thread* principal de l'agent. C'est par ce mécanisme que nous implantons l'autonomie de nos agents.

Pour définir le vecteur *'adresses des paires'*, nous avons créer l'interface *'AgentListener'* qui permet d'étendre l'interface *'EventListener'* de JAVA pour ajouter une méthode *'processEvent()'* qui permet le traitement d'événements. Aussi, chaque élément du vecteur *'adresses des paires'* est un objet d'une classe qui implémente l'interface *'KqmlEvent'* ou d'une de ses sous classes. À son tour la classe *'BasicAgent'* implemente l'interface *'AgentListener'*. La classe *'AgentListener'* fournie aussi deux méthodes *'addListener()'* et *'removeListener()'* pour permettre aux autres agents de s'enregistrer ou se retirer de la liste de distribution de l'agent. Ces deux méthodes supportent aussi les API *'JavaBeans_Event'*, ce qui facilite l'intégration d'objet / agent définit par une classe *'AgentListener'* en utilisant n'importe quel outil qui supporte les *'JavaBeans'*.

La méthode *'Router()'* permet d'implanter le processus de communication de l'agent. En fait, cette méthode définit le traitement par défaut à effectuer lors de la réception de messages (i.e. : *'register'*, *'deny'*, *'sorry'*, *'advertise'*, *'unadvertise'*, *'subscribe'*, *'unsubscribe'*). Pour assurer l'échange de messages sous forme d'événements la méthode *'Router()'* fait appel aux méthodes *'processEvent()'* et *'sendEvent()'*. Ces méthodes lui permettent de faire l'extraction de messages des événements reçus et la construction et l'envoi d'événements à partir de messages. Toutes les trois méthodes sont synchronisées

pour contrôler l'accès aux vecteurs d'adresses des agents dans un environnement *'multithreads'*. Tous les événements manipulés par ces méthodes sont des objets de la classe '*KqmlEvent*'. En fait, cette classe dérive de la classe de JAVA, *'EventObject'*, et nous l'avons introduis pour pouvoir construire des événements complexes. Le constructeur *'KqmlEvent()'* peut prendre comme paramètre soit un seul argument définissant l'agent source de l'événement, soit deux arguments, un pour définir l'agent source et l'autre pour définir l'information accompagnant l'événement. Nous avons défini ce deuxième argument comme un objet pour n'imposer aucune restriction sur la nature de celui-ci. Pour la communication entre les agents nous utilisons cet objet pour envoyer des messages FIPA/KQML. Chaque objet de la classe '*KqmlEvent*' est construit à base d'un objet de la classe '*KqmlObject*'.

La classe *'BasicAgent'* comprend aussi des méthodes pour initialiser l'agent à un état déterminé (*'initAgent()*, *'resetAgent()'*), des méthodes pour démarrer et arrêter l'exécution de l'agent (*'process()'*, *'stop()'*) et une méthode *setSleepTime()* pour générer le temps de pause d'une période d'attente. Pendant le démarrage de l'agent (méthode *'process()'*), les processus correspondant au routeur et à l'initialisation de l'horloge sont lancés. Ainsi, lorsqu'un agent tourne, il y aura au moins deux *threads* en exécution : le *thread* principale où est réalisé le comportement de l'agent et le *thread* du routeur.

Classe KnowledgeBase : cette classe définit l'ensemble des fonctionnalités nécessaires au processus de raisonnement par règles (règles, variables et moteur d'inférence). Parmi ses attributs, la classe '*KnowledgeBase*' déclare une table *'variableList'* contenant toutes les variables utilisées dans la base, un vecteur *'ruleList'* contenant la liste des règles contenues dans la base et des méthodes pour l'implantation du processus d'inférence *'forwardChain()'* et *'backwardChain()'*.

La méthode *'forwardChain()'* implante l'algorithme général du chaînage avant. Au début, l'ensemble des règles candidates est déterminé. Ensuite, ces règles sont triées selon le nombre de clauses dans leur partie condition et une boucle est enclenchée pour les traiter une à une. Après l'exécution de chaque règle, une réévaluation est faite pour

toutes les clauses qui font référence à la variable dont la valeur vient d'être modifiée par la règle. Enfin, lorsque toutes les règles sont exécutées, une réévaluation est faite pour déterminer s'il y a de nouvelles règles candidates parmi celles qui font référence à des variables dont la valeur vient d'être mise à jour par le précédent traitement. La même itération est répétée sur les nouvelles règles candidates.
La méthode *'backwardChain()'* implante l'algorithme général du chaînage arrière. En fait, elle prend comme paramètre une référence au but qu'il faut déterminer, puis cette référence est utilisée pour retrouver la variable de la base qui correspond à ce but. Ensuite, toutes les clauses conséquences qui font référence à cette variable sont déterminées et leurs règles correspondantes deviennent candidates. Pour démontrer chacune de ces clauses conséquences, un appel récursif de la méthode *'backwardChain()'* peut être fait si nécessaire sur les clauses conditions de la règle en cours d'exécution.

Classe Rule : la classe *'Rule'* est utilisée pour définir une règle et comprend aussi certaines variables et méthodes utilisées pour le processus d'inférence. Chaque règle possède un nom, une référence à la base de connaissances à laquelle appartient, un vecteur de clauses conditions et une clause conséquence. La valeur logique de la règle est stockée dans une variable *'truth'*. Nous avons utiliser une déclaration objet pour la variable '*truth*' pour permettre l'utilisation de la valeur *'null'* qui correspond à dire que la vérité de la règle est indéterminée. La variable *'isFired'* indique si cette règle a été déclenchée ou non.
Le constructeur de la classe *'Rule'* prend comme paramètre une référence à une base de connaissance, un nombre variable de références à des clauses conditions et une référence à une clause conséquence. Il initialise les variables *'truth'* à *'null'* et *'isFired'* à *'false'*, puis s'enregistre auprès de sa base de connaissance et ses clauses membres. Ainsi, tout changement dans la valeur des clauses sera automatiquement répercuté sur l'état de la règle.

Classe Clause : une clause est composée d'une variable dans sa partie gauche, d'un opérateur de condition et d'une constante (symbolique ou numérique) dans sa partie droite.
Le constructeur de la classe clause prend comme paramètres, un objet *'kbVariable'*, un objet *'conditionPattern'* et une variable *'String'* pour la constante. Pendant son instantiation, une clause s'enregistre auprès de sa variable *'kbVariable'*, ce qui permet que tout changement dans la valeur de la variable entraîne une réévaluation automatique de la clause.
La classe clause comprend aussi quatre méthodes : la méthode *'addruleRef()'* utilisée par le constructeur de la classe *'Rule'* pour enregistrer la règle auprès de ses clauses membres, la méthode *'check()'* qui permet de tester la valeur de la clause, et la méthode *'getRule()'* qui retourne la référence à la règle qui comprend cette clause.

Classe Variable : nous avons défini cette classe de base variable pour faciliter la manipulation d'une large gamme de valeurs dans le traitement des règles. La classe variable possède comme attributs, un nom qui identifie l'objet variable et une valeur *'String'* qui peut contenir aussi bien un type de base (entier, réel, caractère, ...) qu'un objet dans sa forme la plus générale (*'java.lang.Object'*).
Les deux méthodes *'setValue()'* et *'getValue()'* sont utilisées pour affecter ou ramener la valeur de la variable.

Classe kbVariable : c'est une sous-classe de la classe *'Variable'* et permet d'implanter les fonctions nécessaires pour l'utilisation des variables dans le processus d'inférence. Ainsi, le vecteur *'clauseRefs'* permet de maintenir la liste des clauses qui font référence à la variable, et la méthode *'addClauseRef()'* permet aux clauses de s'enregistrer auprès de la variable. Certaines méthodes de la classe variable sont surchargées et d'autres sont ajoutées. Par exemple, la méthode *'setValue()'* est surchargée de façon à faire un appel à la méthode *'updateClauses()'* après chaque mise à jour de la valeur de la variable. La

méthode *'updateClause()'* permet de réévaluer la vérité des clauses qui font référence à la variable.

Interfaces Sensor et Effector : ces interfaces sont déclarées pour permettre aux règles de faire des appels de fonctions à partir de leurs clauses conditions ou conséquences. Les détecteurs permettent au processus de raisonnement de sortir du moteur d'inférence pour tester certaines conditions dans l'environnement. Les fonctions détecteurs retournent généralement des valeurs booléennes, mais peuvent aussi retourner des données ou des faits à la mémoire de travail. Les actuateurs donnent plus de flexibilité dans la définition des capacités des agents. En fait, un actuateur permet de transformer une règle d'un mécanisme pour générer des faits à un mécanisme pour générer des actions. C'est ce type de règles qui nous permet de définir les actions qu'un agent peut accomplir et auxquelles les autres agents (logiciel / humain) peuvent faire appel en envoyant un message qui déclenche l'action voulue.

Classe Fact : afin d'implanter les faits, nous avons défini la classe *'Fact'* dont le constructeur prend comme paramètre une seule clause. En fait, un fait peut être une affectation d'une valeur à une variable de la base de connaissance, un appel à un actuateur ou un appel à un détecteur.

6.4 Implantation des agents courtiers

Dans cette section, nous présentons les principales fonctionnalités que les agents courtiers doivent implanter pour pouvoir utiliser notre place commerciale virtuelle. Cependant, aucune restriction n'est imposée au niveau de l'architecture ou des outils utilisés pour l'implantation de ces agents. De telles fonctionnalités concernent surtout, les facilités de communication et de négociation nécessaires à l'utilisation des services de la place commerciale et au processus de marchandage. Dans nos tests expérimentaux du modèle de négociation, nous avons utilisé une extension de la classe *'BasicAgent'* pour implanter nos agents courtiers. Cependant, dans le projet MIAMI, différentes plates-formes sont

utilisées pour l'implantation des agents mobiles : GrassHopper [IKV 1997], Voyager [Objectspace 1997].

6.4.1 Agent courtier acheteur

Les principales données manipulées par un agent acheteur sont : (1) sa liste d'achat, un vecteur d'objets décrivant les articles que l'agent désire acheter avec leur prix d'achat, (2) l'inventaire des articles achetés et (3) une table des négociations en cours. La procédure qui permet le déploiement de l'agent dans la place commerciale doit accomplir les tâches suivantes : (1) faire son enregistrement auprès du manager, (2) faire l'initialisation de sa liste d'achat et (3) démarrer le *'thread'* principale de l'agent. Une session de négociation commence lorsque l'agent acheteur sélectionne un article de sa liste d'achat et demande au manager de lui recommander un vendeur potentiel. La demande de recommandation est faite par instantiation d'un objet *'KqmlObject'* dont la performative est *'recommend-one'* qui est envoyé au manager. Le manager répond alors par un message *'Tell'* contenant la référence au vendeur ou par un message *'sorry'*. Ainsi, l'agent acheteur doit être capable de détecter et répondre à plusieurs situations. En effet, si le manager répond à la requête *'recommend-one'* par un message *'tell'* contenant le nom d'un vendeur potentiel, l'agent acheteur doit alors être capable d'entrer en négociation avec ce dernier en lui adressant un message *'CFP'* ou *'propose'*. L'agent acheteur doit aussi être capable de détecter et réagir aux réponses de ses partenaires qui peuvent contenir entre autres des messages de types : *'accept-proposal'*, *'counter-propose'*, *'reject-proposal'* ou *'deny'*. Si la réponse est un *'accept-proposal'*, l'agent doit d'abord répondre par un message *'award-contract'* contenant l'avis de contrat (via le manager), puis faire du ménage dans son état mental en supprimant la session de négociation correspondant au but qui vient d'être satisfait. Si la réponse est un *'reject-proposal'*, l'agent peut soit demander la médiation du manager (cas des agents sociaux), ou tout simplement supprimer la session de négociation et remettre l'article dans sa liste d'achat pour un autre essai ultérieurement. Si la réponse est un *'counter-propose'* l'agent peut soit accepter l'offre et par la suite envoyer une réponse *'award-contract'*, soit générer et envoyer une autre offre, ou éventuellement répéter la même offre que précédemment.

Par rapport à notre stratégie de négociation seul l'agent acheteur peut répéter la même offre.

6.4.2 Agent courtier vendeur

Les principales données manipulées par un agent vendeur comprennent, une liste d'articles à vendre et une table des négociations en cours. L'agent acheteur doit s'enregistrer auprès du manager avant de pouvoir faire des interactions dans la place commerciale. En plus d'enregistrer l'agent auprès du manager, la fonction de déploiement de l'agent doit aussi faire l'initialisation de l'inventaire des articles à vendre et faire l'enregistrement de l'agent auprès des communautés de son intérêt en envoyant des messages *'advertise'* au manager.

Dans son interaction avec des acheteurs, l'agent vendeur doit pouvoir détecter et traiter deux principales cas : (1) répondre à un appel d'offre initial *'CFP'* (Call For Proposal) ou (2) répondre à une offre se rapportant à une négociation en cours *'counter-propose'*. Dans le premier cas, l'agent doit d'abord vérifier s'il dispose des articles spécifiés dans le *'CFP'*, retirer ces derniers de son inventaire et initialiser une session de négociation en envoyant un message *'propose'* à l'agent acheteur. Dans le second cas, il existe une session de négociation qui est en cours. Ainsi l'agent utilise l'identifiant envoyé avec le message *'counter-propose'* pour retrouver la session correspondante, puis construire et envoyer une réponse. Le message réponse peut être, soit une re-formulation d'une autre offre *'propose'* en utilisant éventuellement sa stratégie de négociation, soit le rejet de l'offre *'reject-proposal'* et par suite l'arrêt de la négociation. Si aucune session de négociation n'a pu être identifiée, alors les articles en question viennent d'être vendus et l'agent doit répondre à l'offre reçue par un message *'Deny'*, puis supprimer la session de négociation correspondante. L'agent vendeur peut aussi recevoir un message contenant l'avis de contrat '*award-contract*' si son offre est acceptée. Dans ce cas l'agent doit retirer de son inventaire les articles vendus et terminer la session de négociation correspondante.

Plusieurs sophistications dans le raisonnement ou l'état mental des agents peuvent être implantées selon le choix du développeur. Aussi, d'autres services sont fournis par la place commerciale, comme la construction de solution initiale ou la médiation de conflits, mais qui nécessitent des agents clients de supporter d'autres types de messages. En fait, la construction de proposition initiale nécessite de l'agent de pouvoir construire et envoyer au manager des messages de type *'ask'* concernant des articles donnés et d'interpréter la réponse de ce dernier qui vient sous forme d'un message *'reply'*. La médiation nécessite de l'agent de supporter le traitement des messages de type : *'mediate'*, *'mutual-propose'*, *'multi-propose'* et *'joint-propose'*.

Dans notre implantation des agents courtiers, nous avons distingué trois types d'agents (cf. 7) : agents primaires (classe *'Broker'*), agents rationnels (classe *'RationalBroker'*) et agents sociaux (classe *'SocialBroker'*). Les agents primaires implantent seulement les fonctionnalités que nous avons définies dans les sections 4.1. et 4.2.. Les agents rationnels implantent, en plus des fonctionnalités des agents primaires, des facilités relatives à l'utilisation de la stratégie de négociation à base de risque. Les agents sociaux implantent, en plus des fonctionnalités des agents rationnels, des facilités relatives à la médiation et à l'utilisation des relations de dépendances pour la construction de différents types de contrats : contrats par groupement, contrats par échange ou contrats multiagent.

Chapitre 7: Résultats de simulation

L'efficacité d'un modèle de négociation peut être mesurée aussi bien de façon qualitative (qualité de la solution produite), que quantitative (coût du cycle de négociation en temps et ressources). Les mesures peuvent être prises et traitées à deux niveaux : niveau individu et niveau société. Ainsi, avant de pouvoir dire que la stratégie de négociation et le modèle de médiation proposés dans cette thèse répondent bien aux exigences de la plate-forme MIAMI, un certain nombre de questions importantes nécessitent une réponse.

- Quelles performances réalisent les agents qui implantent un tel modèle, en terme de profit, satisfaction de buts et charge de communication ?
- À quel niveau sont situés ces agents, si on les compare dans une perspective globale du système aux autres types d'agents ?
- Quels sont les bénéfices ou effets indésirables engendrés par l'utilisation de la stratégie de négociation et des relations de dépendances ?

Pour répondre à ces questions et faire la validation du modèle présenté ci-dessus, nous avons implanté trois types d'agents et avons défini deux types de métriques.

Les trois types d'agents implantés sont :

- ***agents primaires*** : ce sont des agents qui n'acceptent que des offres qui leurs sont individuellement rationnelles. Cependant, ils n'utilisent ni stratégie de négociation ni relations de dépendances dans leurs processus de négociation.
- ***agents rationnels*** : ce sont des agents qui implantent la stratégie de négociation à base de risque (cf. 5.4), mais n'utilisent pas les relations de dépendances.
- ***agents sociaux*** : ce sont des agents qui implantent la stratégie de négociation et les relations de dépendances. En fait, ils utilisent la stratégie de négociation à base de risque pour générer leurs offres et exploitent les relations de dépendances, qu'ils ont avec leurs partenaires, pour générer différents types de contrats qui aident à résoudre les conflits.

Pour évaluer le modèle nous avons considéré deux types de métriques : certaines pour mesurer la qualité des solutions produites et d'autres pour mesurer l'efficacité du modèle. Ainsi, pour la qualité des solutions, nous avons pris des mesures sur les profits[44] réalisés par les agents et, pour l'efficacité du modèle, nous avons pris des mesures sur le nombre de buts satisfaits et sur la charge de communication.

- ***Profit individuel (PI)*** : le profit individuel est défini pour un agent vendeur par la différence entre le prix du contrat et le prix minimal acceptable par l'agent, et pour un agent acheteur par la différence entre le prix maximal acceptable par l'agent et le prix du contrat.
- ***Profit conjoint (PC)*** : le profit conjoint est calculé par la combinaison des profits individuels des agents participant dans le contrat. Si N agents participent dans un contrat alors $PC = \Sigma^{N}_{i=1} PI_i$, où PI_i représente le profit individuel de l'agent A_i.
- ***Vitesse de satisfaction des buts*** : la vitesse de satisfaction des buts est calculée par la distribution du nombre cumulé de buts satisfaits sur la durée de la simulation (mesurée en nombre de buts par seconde).
- ***Charge de communication*** : la charge de communication est mesurée par le nombre cumulé de messages contenant des performatives de communication traitées (envoyés ou reçus) par l'agent. Ainsi, seuls les messages de type, *« propose, counter-propose, CFP, ask, mediate, reply, ... »* sont considérés, et les messages de type *« Register, Subscribe, Advertise, ... »* ne le sont pas.

Afin de répondre aux questions posées ci-dessus, nous avons mené plusieurs essais expérimentaux. Dans ces essais, deux principales dimensions sont variées : (1) le type d'agents (Agent primaire, rationnel, social) et (2) le type de partenaires avec lesquels l'agent interagit.

Pour permettre une comparaison entre les différentes situations, nous avons placé les trois types d'agents dans des contextes de simulations identiques. Pour cela, les données de simulation ont été produites par des générateurs aléatoires de distribution uniforme et stockées dans des fichiers. Pour chaque agent activé pendant la simulation, nous avons

[44] Dans ce contexte, le terme «profit» désigne aussi bien une valeur d'utilité positive que négative ou nulle.

généré un ensemble de 200 buts, chaque but consiste en un service à vendre ou à acheter. Pour chaque but sont spécifiés, ses caractéristiques, le prix minimal acceptable pour la vente ou le prix maximal acceptable pour l'achat. Ainsi, chaque agent possède deux fichiers pour ces buts : sa liste d'achat et son inventaire de vente. Dans chaque expérience nous avons identifié deux types d'intervenants, l'agent dont la performance est mesurée selon les différentes métriques définies ci-dessus et le reste de la société (un nombre arbitraire de partenaires de même type avec lesquels l'agent interagit). Tous les agents utilisés pour les simulations ont été implantés selon la même architecture de base définie dans la section 6.4. chaque session de simulation prend une durée de 20 minutes.

Pendant la simulation, les agents déterminent leurs buts candidats en les sélectionnant de leurs listes d'achat ou en recevant un appel d'offre relatif à des articles qu'ils ont dans leur inventaire. Les données relatives aux paramètres que nous avons mesurés pendant la simulation ont été stockées dans des fichiers, puis analysées ultérieurement à l'aide du logiciel MatLab [Hanselman 1996]. Les sections suivantes présentent l'analyse de ces résultats.

L'annexe III décrit le déroulement d'un cycle de négociation en utilisant un exemple numérique. L'importance de l'utilisation des dépendances et des différents types de contrats est mise en évidence.

7.1 Analyse de la performance par type d'agents

Agent primaire: dans les figures[45] 7.1, 7.2 et 7.3, on considère la performance réalisée par un agent primaire ayant toujours le même ensemble de buts à satisfaire mais en faisant varier le type de ses partenaires. Ainsi, pour son profit individuel, la figure 7.1 montre que *l'agent primaire* réalise sa meilleure performance face à des *agents sociaux*, soit le triple du profit réalisé face à des *agents primaires ou rationnels*. Une telle performance ne peut être expliquée que par l'effet social des relations de dépendances utilisées par les partenaires qui sont de type agents sociaux. Dans le premier segment des

[45] Tout le long de cette section, même si les graphiques regroupent plusieurs courbes, ces derniers correspondent à des simulations différentes mais réalisées avec les mêmes données et dans le même contexte.

courbes de la figure 7.1, on peut voir un léger avantage sur la vitesse de réalisation de profit par l'agent lorsque ces partenaires sont des agents primaires ; cela est dû au comportement réactif de ces derniers. Cependant, si on considère toute la période de simulation, le taux moyen de croissance du profit reste largement supérieur face à des partenaires de type social soit (11.04 $ /sec) contre (8.56 $ /sec) pour les partenaires de type primaire et (8.25$ /sec) pour les partenaires de type rationnel.

En ce qui concerne la satisfaction de buts, la figure 7.2 montre que l'agent réalise aussi sa meilleure performance face à des agents sociaux soit 47%[46] de plus qu'avec des partenaires primaires et 55% de plus qu'avec des partenaires rationnels.

La performance de l'agent primaire sur la charge de communication est pratiquement la même face à tous les types de partenaires (figure 7.3).

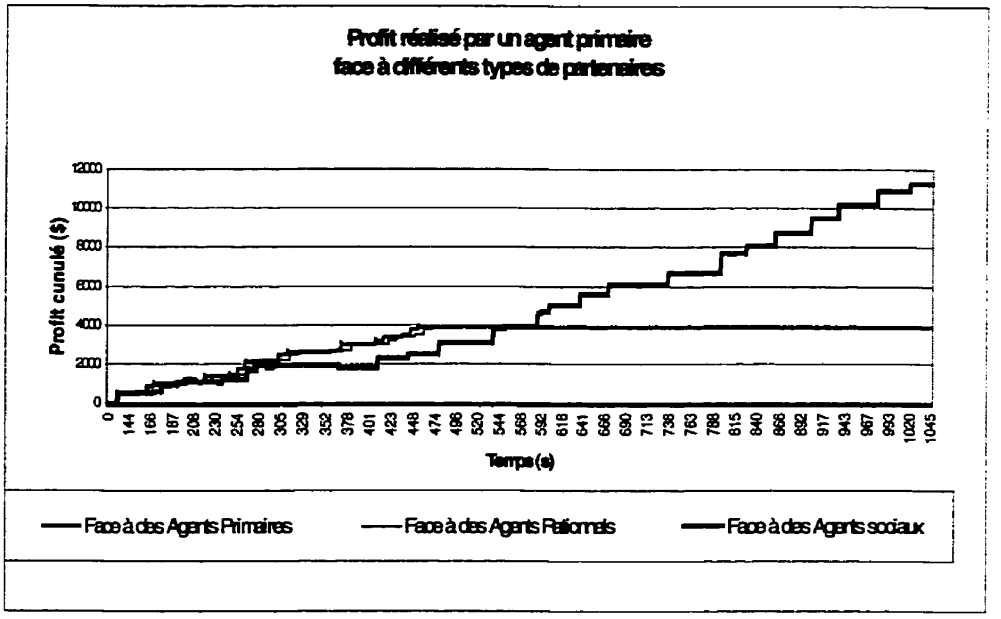


Figure 7.1: Profit réalisé par un agent primaire selon différents types de partenaires

[46] Durant l'analyse de ces résultats de simulation, quand une performance est donnée en pourcentage sans indiquer la référence à laquelle se rapporte, alors c'est la performance de l'agent de type primaire qui est considérée comme référence de base.

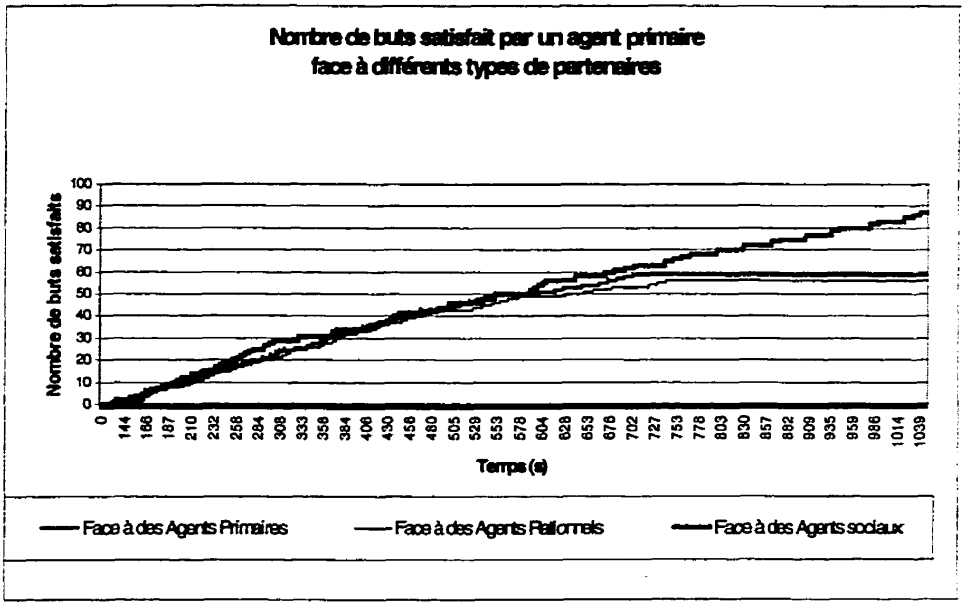


Figure 7.2: Buts satisfaits par un agent primaire selon différents types de partenaires

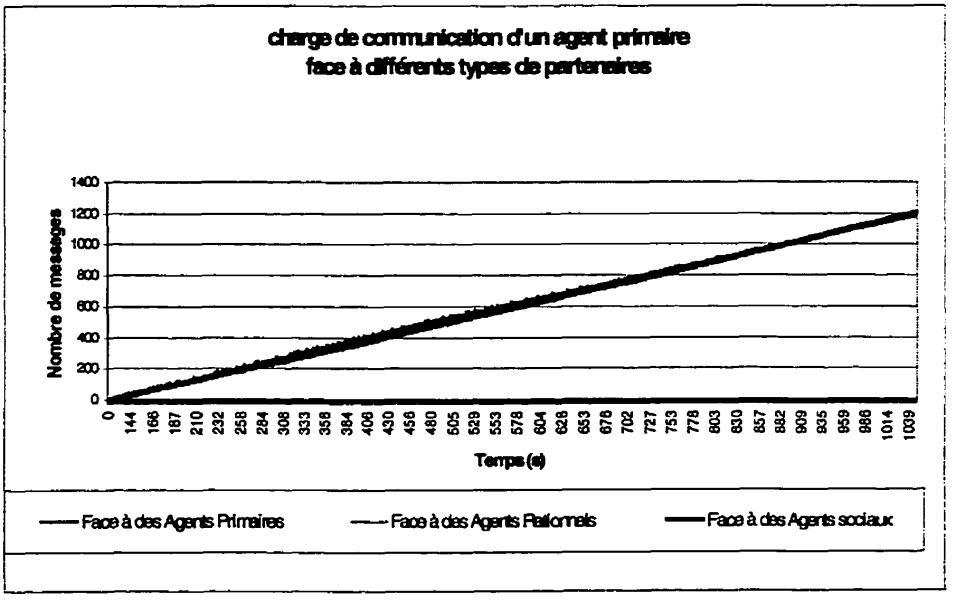


Figure 7.3: Charge de communication d'un agent primaire selon différents types de partenaires

Agent rationnel : l'agent rationnel réalise son meilleur profit face à des agents sociaux (voir la figure 7.4), soit 40% de plus que le profit réalisé lorsque ses partenaires sont de type rationnel et 35% de plus lorsque ses partenaires sont de type primaire. Cependant, en ce qui concerne la vitesse de réalisation du profit on peut voir, surtout dans le segment constituant les premiers 2/3 de la période de simulation (figure 7.4), que la négociation de l'agent rationnel[47] face à des agents de type sociaux, est pénalisée par l'effet social des relations de dépendances. En effet, l'agent ne réalise qu'un taux de croissance sur le profit de 9.13$/sec face à des partenaires de type social, contrairement à un taux de 10.22$/sec face à des partenaires de type primaire et un taux de 9.15$/sec face à des partenaires de type rationnel. Toutefois, l'agent rationnel récupère à la fin sur le profit à réaliser face à des agents sociaux grâce au supplément de buts qu'il va arriver à satisfaire avec ses derniers. Dans ce comportement de l'agent rationnel face aux agents sociaux, nous pouvons déjà voir l'équilibre qui commence à émerger entre, d'une part, l'intérêt individuel de l'agent, représenté par les fonctions d'utilités de la stratégie de négociation et, d'autre part, l'intérêt global du système représenté par l'effet social des relations de dépendances.

En ce qui concerne la satisfaction de but, la figure 7.5 montre bien que la performance de l'agent rationnel face à des agents sociaux est supérieure à sa performance face aux autres types d'agents, et cela, tant pour la vitesse de satisfaction des buts que pour le nombre total de buts satisfaits, soit 53% de plus que le nombre de buts satisfaits face à des agents primaires et 67% de plus que le nombre de buts satisfaits face à des agents rationnels.

La charge de communication est quasi similaire face aux agents primaires et rationnels, par contre elle est réduite de 17% face aux agents sociaux (figure 7.6).

[47] Il faut noter que les agents de type rationnel n'utilisent que la stratégie de négociation basée sur l'évaluation du risque, et de ce fait ne s'intéressent qu'à leur profits individuels.

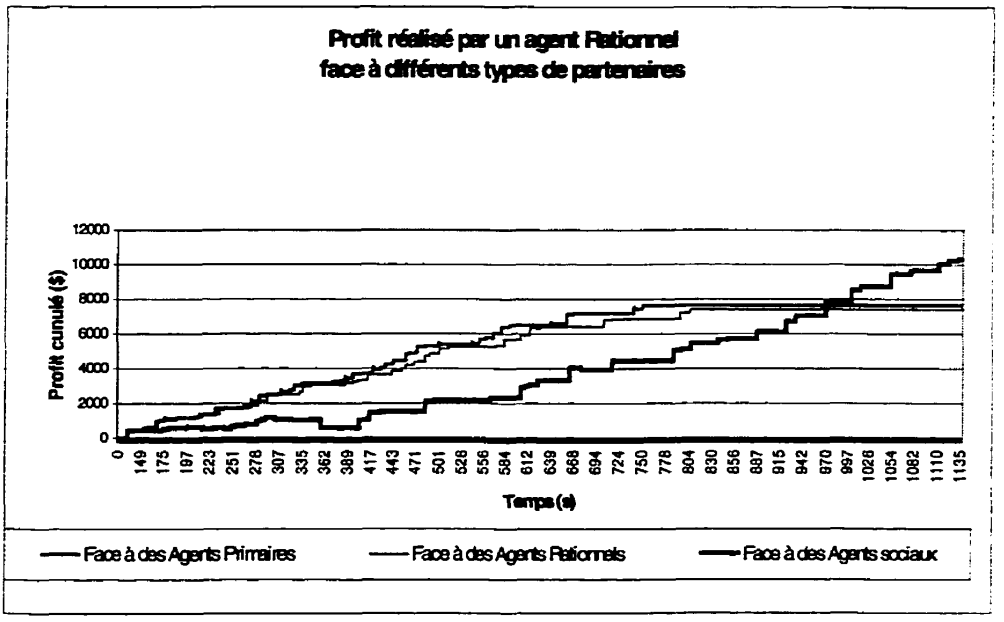


Figure 7.4: Profit réalisé par un agent rationnel selon différents types de partenaires

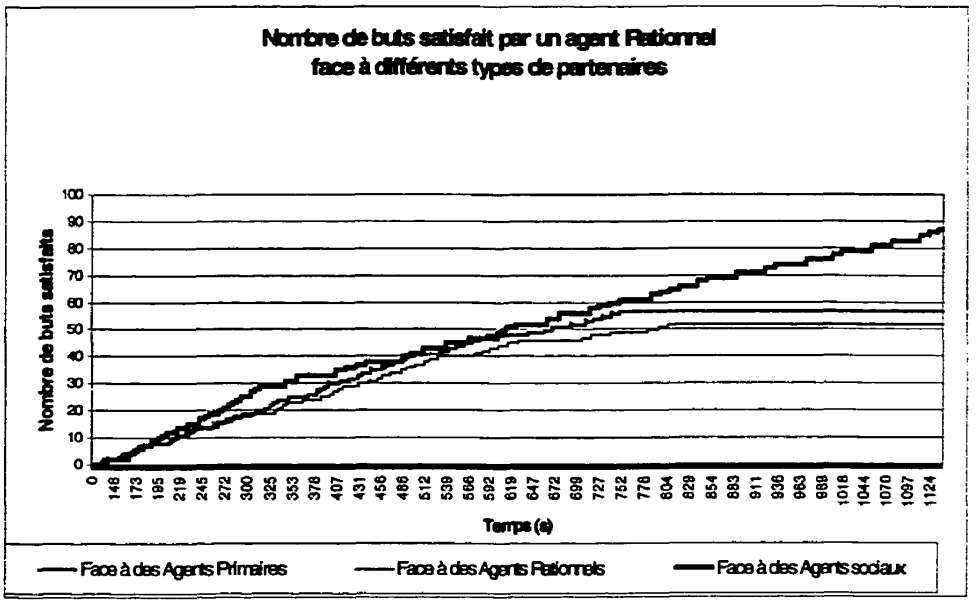


Figure 7.5: Buts satisfaits par un agent rationnel selon différents types de partenaires

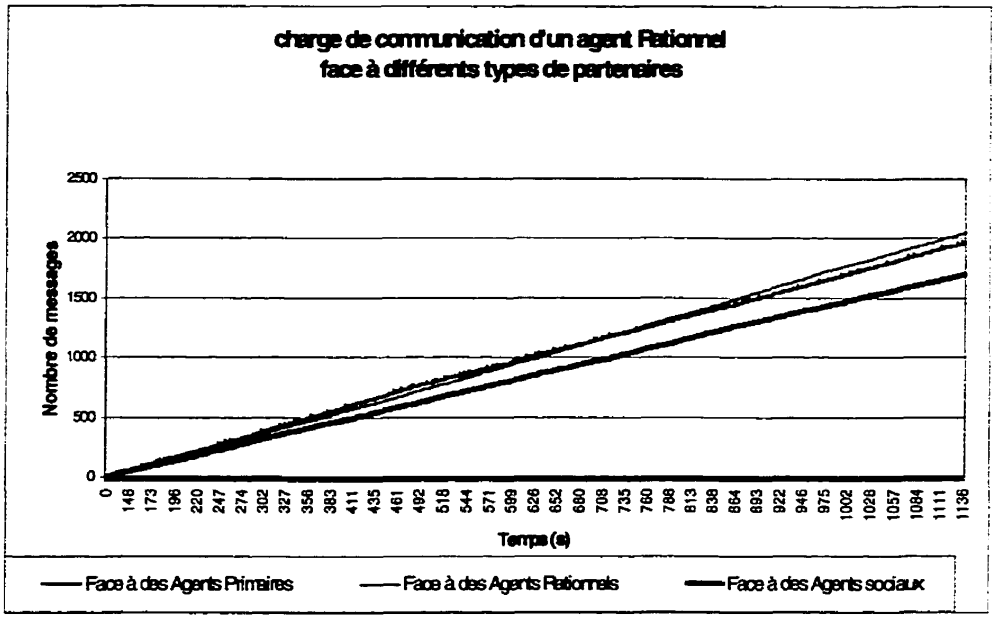


Figure 7.6: Charge de communication d'un agent rationnel selon différents types de partenaires

Agent social : comme tous les autres agents, l'agent social réalise son meilleur profit face à des partenaires de type sociaux (figure 7.7), soit 47% de plus que le profit réalisé face à des agents primaires et 83% de plus que le profit réalisé face à des agents rationnels. Même si les premières parties des courbes de la figure 7.7 montrent un léger avantage sur la vitesse de réalisation du profit lorsque l'agent négocie avec des agents primaires, le meilleur taux moyen de croissance est réalisé avec des partenaires de type social. En fait, ce taux est de 9.44$/sec avec des partenaires sociaux, contre seulement 7.60$/sec avec des partenaires primaires et 5.18$/sec avec des partenaires rationnels.

Quant à la satisfaction des buts, l'agent social réalise sa meilleure performance face à des agents sociaux (figure 7.8), soit 23 % de plus que le nombre de buts satisfaits face à des agents primaires et 31% de plus que le nombre de buts satisfaits face à des agents rationnels.

La charge de communication est en moyenne quasi similaire face aux trois types d'agents (figure 7.9).

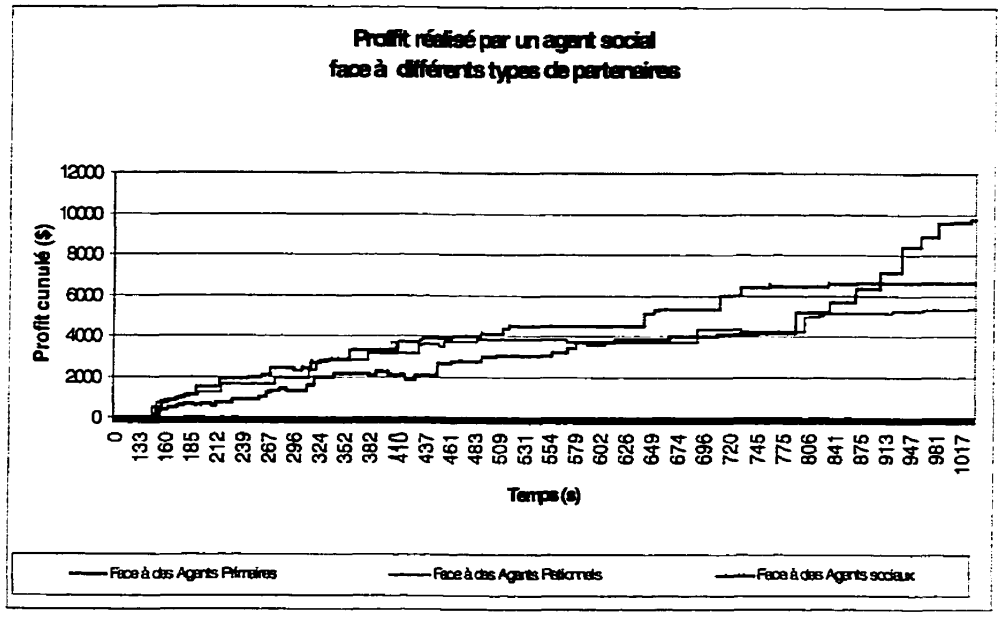


Figure 7.7: Profit réalisé par un agent social selon différents types de partenaires

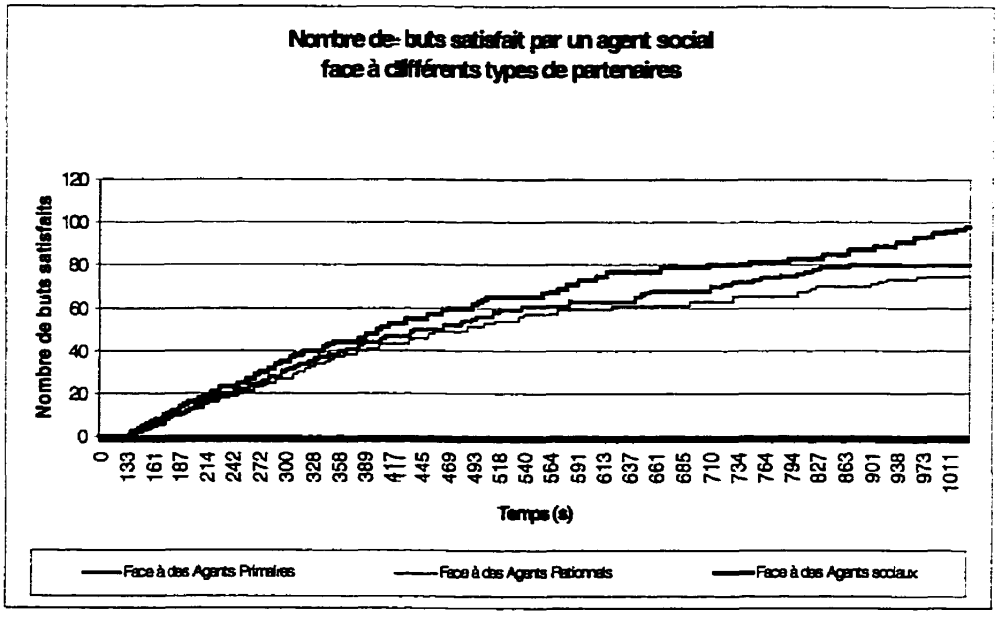


Figure 7.8: Buts satisfaits par un agent social selon différents types de partenaires

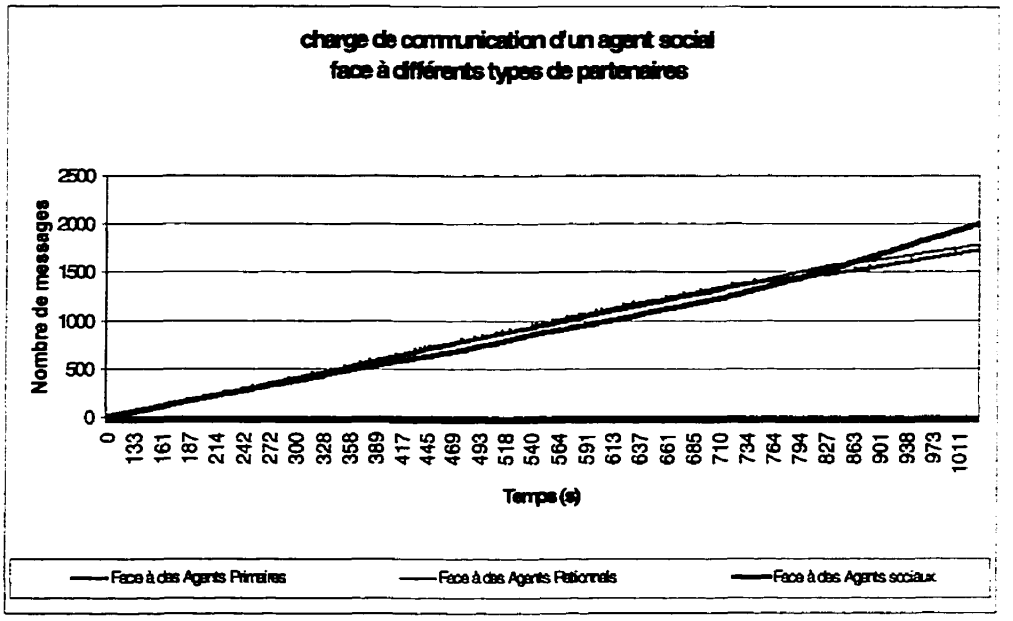


Figure 7.9: Charge de communication d'un agent social selon différents types de partenaires

7.2 Comparaison de la performance des agents

Les résultats des simulations montrent que l'utilisation de la stratégie de négociation permet aux agents rationnels de doubler leur profit dans les cas où leurs partenaires sont des agents primaires ou rationnels (figures 7.10 et 7.11). Cependant, pour le cas où leurs partenaires sont de type social (figure 7.12), on remarque que l'utilisation des relations de dépendances améliore le profit des agents primaires qui réalisent la meilleure performance, soit en moyenne 19% de plus que les agents sociaux et 9% de plus que les agents rationnels.

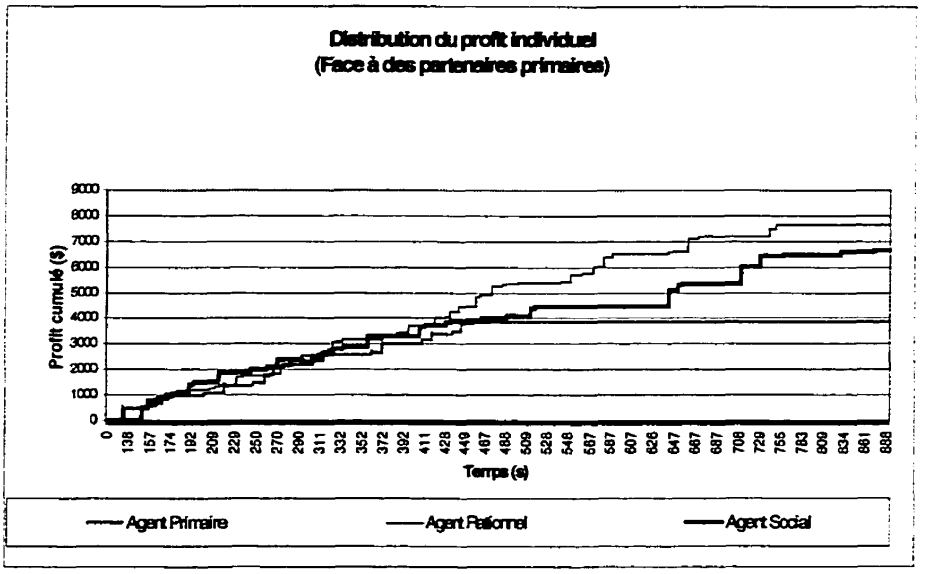


Figure 7.10: Comparaison des profits réalisés par chaque type d'agent face à des négociateurs de type primaire

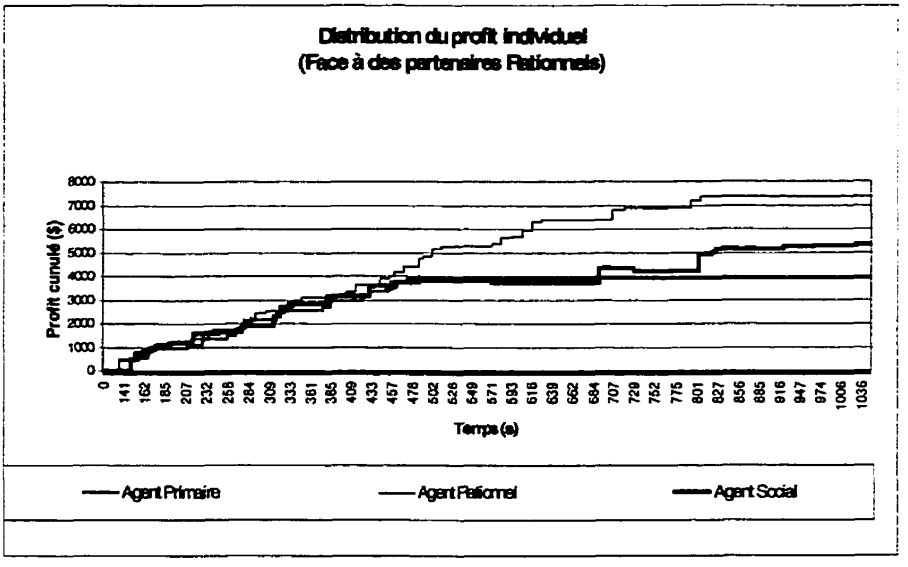


Figure 7.11: Comparaison des profits réalisés par chaque type d'agent face à des négociateurs de type rationnel

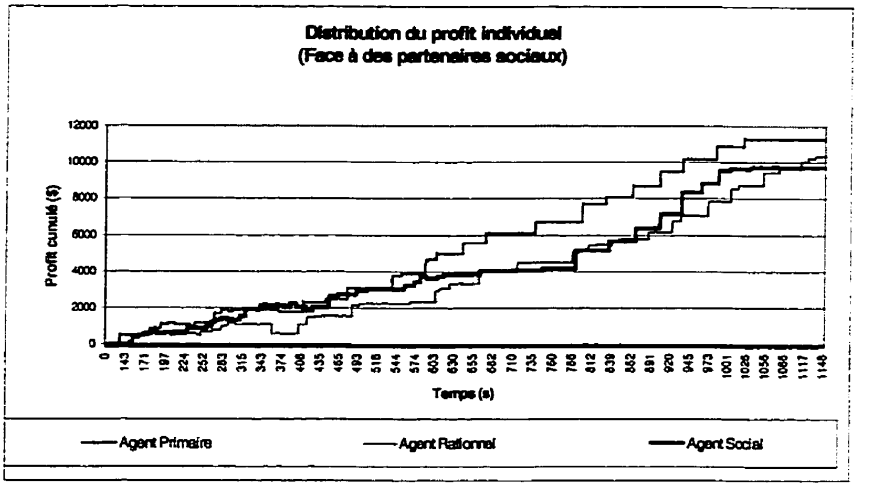


Figure 7.12: Comparaison des profits réalisés par chaque type d'agent face à des négociateurs de type social

Si on considère le paramètre de satisfaction de but, on peut voir (figures 7.13, 7.14, 7.15) que l'utilisation des relations de dépendances permet aux agents sociaux de réaliser la meilleure performance dans tous les cas.

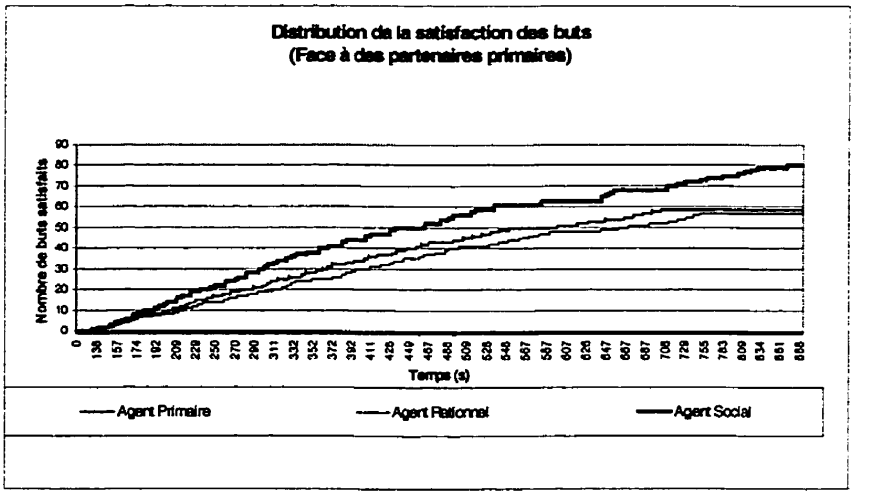


Figure 7.13: Comparaison du nombre de buts satisfaits par chaque type d'agent face à des négociateurs de type primaire

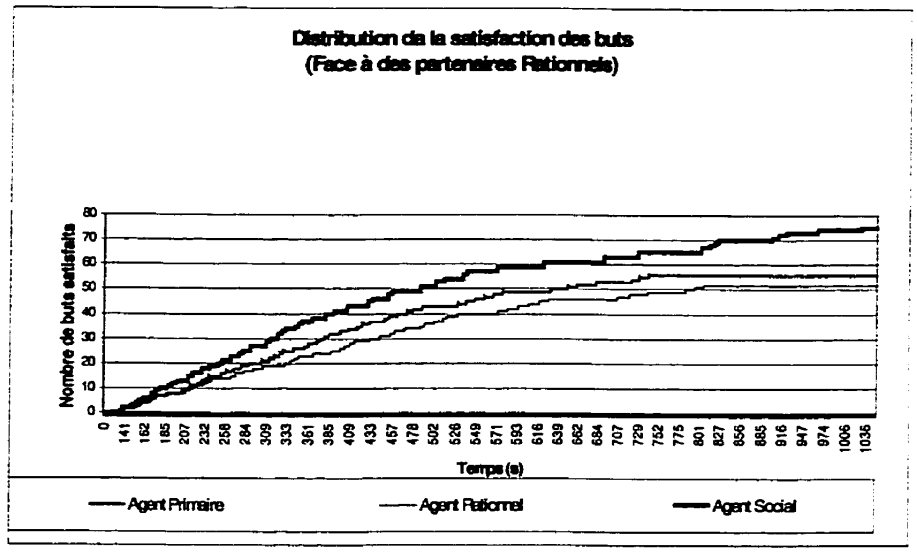


Figure 7.14: Comparaison du nombre de buts satisfaits par chaque type d'agent face à des négociateurs de type rationnel

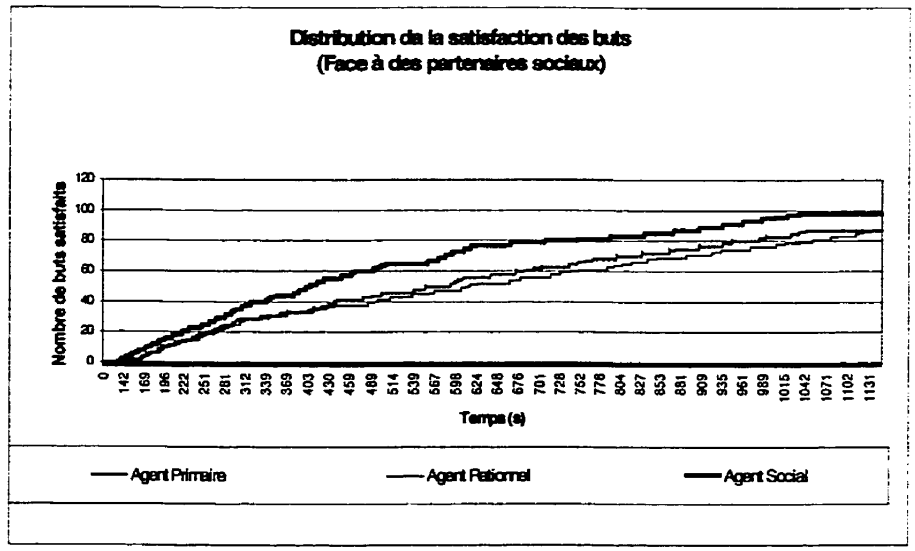


Figure 7.15: Comparaison du nombre de buts satisfaits par chaque type d'agent face à des négociateurs de type social

Si on fait une analyse des résultats moyens des différentes simulations réalisées en faisant varier les types des agents, on peut voir (figure 7.16) que l'utilisation de la stratégie de négociation basée sur l'évaluation de risque permet d'augmenter le profit de 33%, mais aussi entraîne une réduction sur la satisfaction de buts de 3% (agents rationnels vs agents primaires). Cependant, lorsqu'on combine la stratégie de négociation avec les relations de dépendances, on obtient une augmentation sur le profit de 15% ainsi qu'une augmentation sur la satisfaction de buts de 25% (agents sociaux vs agents primaires). En fait, l'utilisation des relations de dépendances dans le processus de médiation nous permet de chercher un équilibre entre la réalisation de profit individuel et la réalisation de buts. En d'autres termes, faire l'équilibre entre (1) servir l'objectif des agents individuels qui consiste à maximiser leur profit et (2) permettre au système d'opérer d'une façon cohérente en satisfaisant plus de buts à différents niveaux.

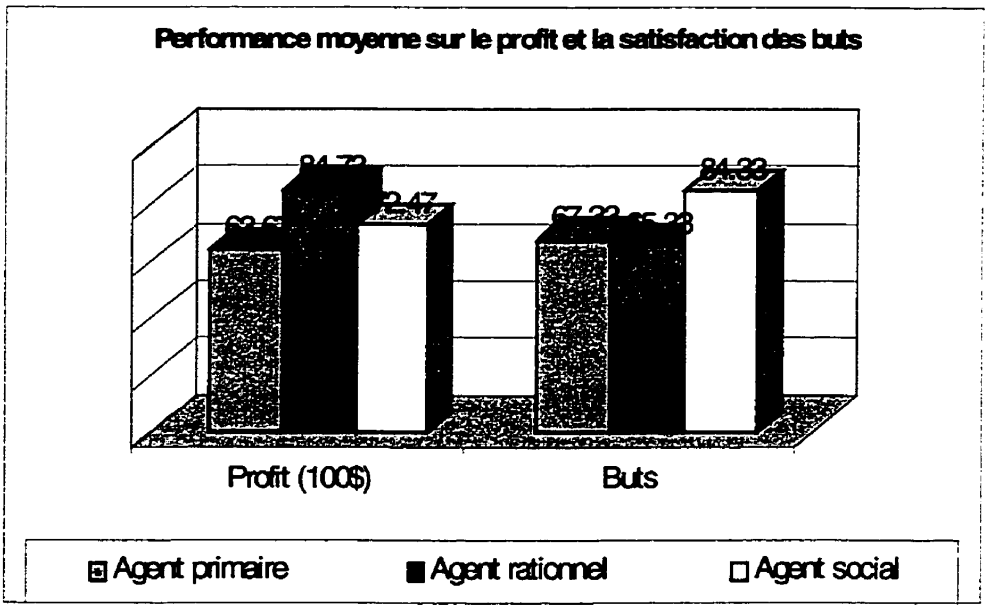


Figure 7.16: Performance moyenne des agents sur le profit individuel et la satisfaction de buts

Le principe de chercher un équilibre entre la vision centrée sur l'autonomie des agents et la vision centrée sur le comportement social du système global, nous conduit à faire une

comparaison avec la notion de ***rationalité sociale***[48] définit par N. R. Jennings & al. [Jennings 1997, Hogg 1997, Kalenka 1999], où les agents travaillent pour la maximisation du profit social du système. Par conséquent, ces derniers acceptent tout contrat dont le profit conjoint est positif même si ce contrat n'est pas individuellement rationnel pour certains agents participants.
Dans ce travail, nous avons considéré la notion ***d'engagement social*** [Esmahi 1999c, Esmahi 2000b] que nous définissons par :

Principe d'engagement social *: si un agent a à choisir entre différents contrats individuellement rationnels, alors il choisit celui qui permet de maximiser le profit conjoint des agents participants et qui minimise l'écart dans la distribution du profit entre ces agents.*

Nous n'avons implanté aucun mécanisme explicite pour garantir le principe d'engagement social. Cependant, nos agents sociaux adhèrent bien à ce principe. En effet, l'utilisation des fonctions d'utilités dans la stratégie de négociation (cf. 5.4) empêche les agents d'accepter des contrats non rationnels, et l'introduction des relations de dépendances et de différents types de contrats (cf. 5.3.2) dans le processus de médiation permet à ces agents de maximiser le profit conjoint (figure 7.17) et de minimiser l'écart entre les profits des agents (figure 7.18).
Ainsi, la figure 7.17 montre que le profit conjoint réalisé dans des négociations qui impliquent des agents de type social est en moyenne deux fois plus grand que le profit conjoint réalisé dans des négociations qui n'impliquent que des agents de type primaire ou rationnel. Aussi on peut voir dans la figure 7.18 que l'écart de profit est à son plus bas niveau dans des négociations qui impliquent des agents de type social, soit moins de la moitié de l'écart de profit réalisé par des négociations qui n'impliquent que des agents de type primaire ou rationnel.

---

[48] Le principe de rationalité sociale a été défini par N.R. Jennings et al. par : « *if a socially rational agent can perform an action whose joint benefit is greater than its joint loss, then it may select that action* ».

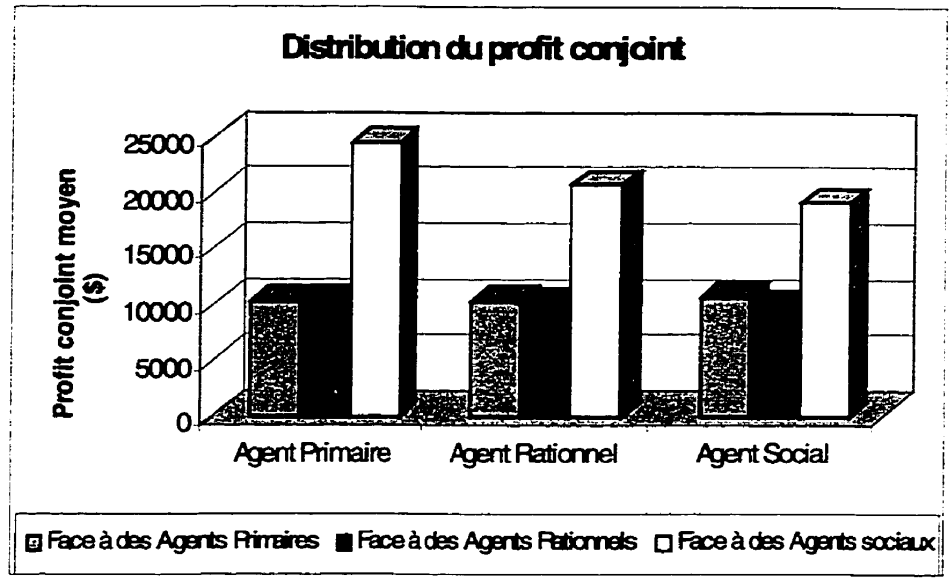


Figure 7.17: Distribution du profit conjoint moyen réalisé par chaque type d'agent selon le type de ses partenaires

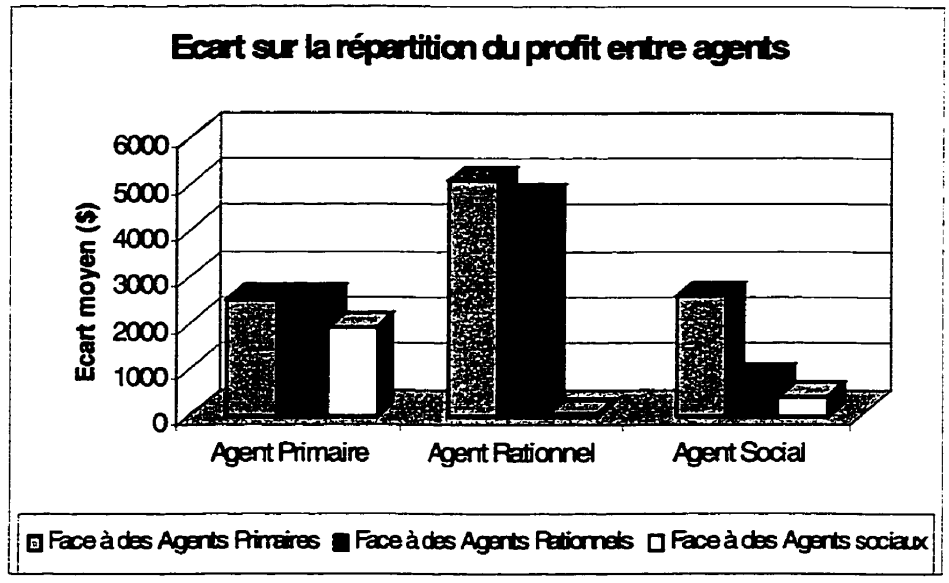


Figure 7.18: Écart moyen dans la répartition des profits entres les agents

L'utilisation de la stratégie de négociation basée sur le calcul du risque et du protocole de négociation où les agents peuvent s'échanger plusieurs propositions et contre-propositions, entraîne une augmentation moyenne de 20% de la charge de communication (figure 7.19, 7.20, 7.21). Pour mieux expliquer cette augmentation dans la charge de communication, nous allons la relativiser par rapport au nombre de buts satisfaits. En effet le calcul du taux moyen de messages par but nous donne les résultats du tableau 7.1.

Tableau 7.1: Taux moyen de messages par but

	Agent Primaire	Agent Rationnel	Agent Social
Face à des agents primaires	13.12	21.72	19.13
Face à des agents rationnels	14.80	26.10	23.56
Face à des agents sociaux	13.78	19.46	20.31
Moyenne	13.90	22.43	21.00
Augmentation moyenne par rapport à l'agent primaire		61 %	51 %

Ainsi, l'augmentation de 61% sur le nombre moyen de messages par but chez les agents rationnels ne peut être attribuée qu'à l'utilisation de la stratégie de négociation, qui est la seule variante qui les différencie des agents primaires. Aussi, la diminution de cette augmentation de 10% chez les agents sociaux ne peut être attribuée qu'à l'utilisation des relations de dépendances qui sont la seule variante qui les différencie des agents rationnels.

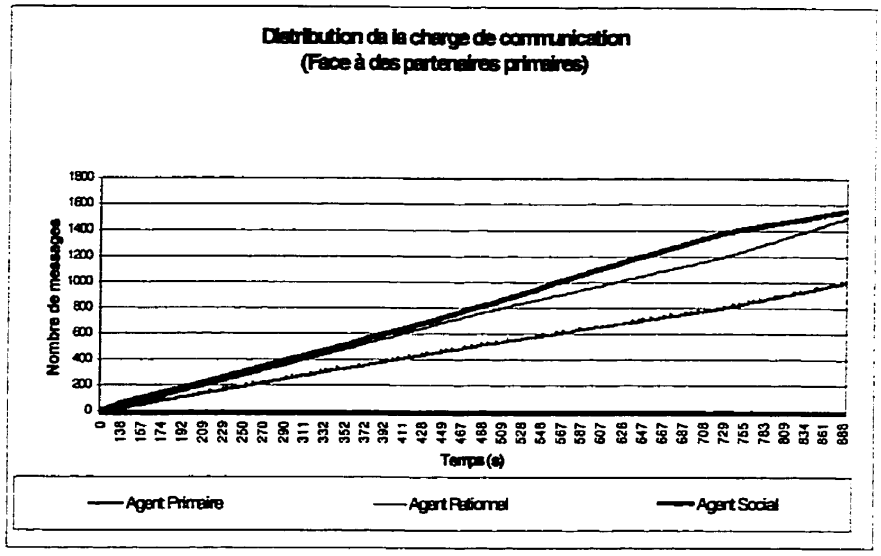


Figure 7.19: Comparaison de la charge de communication de chaque type d'agent face à des négociateurs de type primaire

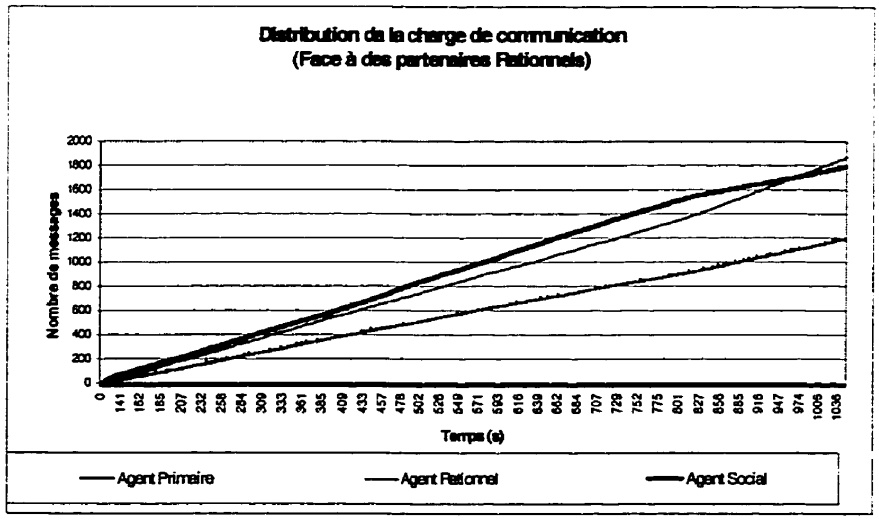


Figure 7.20: Comparaison de la charge de communication de chaque type d'agent face à des négociateurs de type rationnel

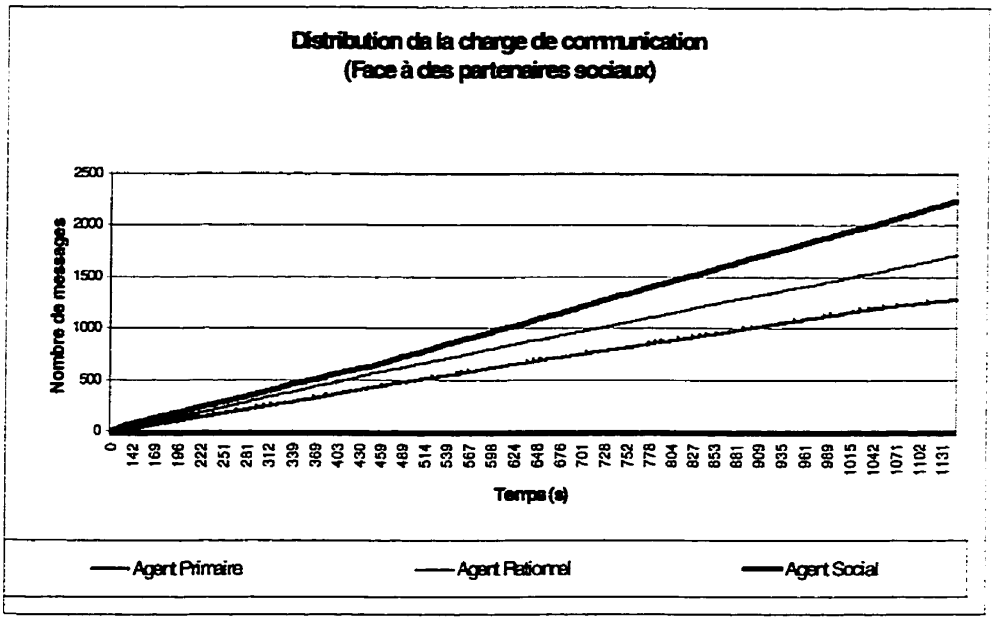


Figure 7.21: Comparaison de la charge de communication de chaque type d'agent face à des négociateurs de type social

Chapitre 8: Conclusion

Dans cette thèse, nous avons proposé un modèle de négociation fondé sur la persuasion en utilisant les relations de dépendances et le raisonnement par cas. Nous avons montré que ce modèle permet de faire un équilibre entre le besoin de rationalité individuelle des agents et le besoin de cohérence sociale du système. Particulièrement, nous avons appelé *principe d'engagement social*, le principe par lequel des agents autonomes et individuellement rationnels prennent aussi en charge l'équilibre social du système global.

À travers une analyse de la problématique de la coopération et des interactions dans les SMA, nous avons montré l'importance de prendre en considération les relations bilatérales entre les agents pour construire un modèle de négociation qui assure un tel équilibre social. En fait, pour être utilisé dans un contexte pratique, un mécanisme de raisonnement social doit intégrer une notion sur les limites des agents. D'ailleurs, être social et prendre en compte la présence des autres agents est en soi une limite pour les agents.

Aussi, par le modèle de négociation proposé, nous avons montré qu'une façon pour gérer de telles limites est de donner la priorité aux relations bilatérales entre les agents dans le processus de médiation des conflits. Nous croyons que cette façon permet aux agents de bien exploiter leurs ressources et de s'efforcer à faire l'équilibre entre leurs buts individuels et l'objectif global du système.

Pour la construction de ce modèle hybride, nous avons utilisé plusieurs théories : les fonctions d'utilités, les relations de dépendances et la persuasion. Pour la construction et l'évaluation de propositions, nous avons défini une stratégie de négociation basée sur les fonctions d'utilité et le calcul de risque et, pour persuader les autres agents sur une proposition, nous avons défini des procédures pour générer différents types d'arguments.

Pour valider le modèle proposé, nous avons implanté des prototypes sur les plates-formes MIAMI et LALO. Principalement, la place commerciale virtuelle MIAMAP qui implante une grande partie du modèle, nous a permis de faire des évaluations expérimentales. Dans

notre implantation du modèle, nous ne supposons pas que les agents soient homogènes en ce qui concerne leur architecture, outil d'implantation ou mécanismes internes. Seules des facilités de communication supportées par un langage commun sont exigibles.
Les résultats expérimentaux ont montré l'intérêt du modèle aussi bien pour l'augmentation du profit individuel des agents que pour l'amélioration du profit global du système et du nombre de buts satisfaits. En fait, la conséquence la plus évidente de l'utilisation de ce modèle dans MIAMAP, est sa capacité d'améliorer nettement *l'engagement social* des agents. Ainsi, les agents génèrent des contrats qui satisfont plus de buts dans les trois domaines de MIAMI et qui maximisent le bien être social du système tout en minimisant l'écart dans la distribution des profits.

8.1 Quelques limites et remarques sur le modèle proposé

Le présent paragraphe présente quelques limites qui doivent être prises en considération pour l'implantation du modèle. Ces limites sont généralement liées à la rationalité limitée des agents.

a- Implications sur les limites computationnelles des agents

L'instanciation de plusieurs paramètres dans le modèle est laissée au développeur ou à l'utilisateur. Ces paramètres concernent surtout les limites computationnelles des agents. En effet, le développeur ou l'utilisateur doit spécifier le degré de raffinage d'une proposition en fonction de l'incertitude qu'il accepte dans l'évaluation de ces utilités ou son facteur de risque, aussi il doit décider du temps à accorder pour un cycle de négociation ou de médiation.

b- Planification de la délibération locale

Déterminer quand terminer une recherche dans un système distribué est une tâche difficile, surtout avec des algorithmes itératifs distribués qui ne sont pas basés sur un schéma systématique avec des retours arrière. Un agent doit décider du temps de traitement à accorder au raffinage de l'estimation de ses profits et si l'agent est engagé dans plusieurs négociations, il doit aussi décider sur quel ensemble de tâches il doit focaliser son traitement. Aussi, l'agent doit décider selon quelle séquence il va traiter les

différents engagements pris. Dans notre implantation, toutes les tâches sont exécutées selon l'ordre chronologique des engagements et nous n'avons rien proposé pour l'ordonnancement et la planification des tâches des agents.

c- Engagement dans des négociations multiples et simultanées

Afin d'accélérer le processus de négociation, le modèle permet à un agent de s'engager dans plusieurs négociations simultanées. Ainsi, l'agent doit décider du temps qu'il puisse consacrer à l'évaluation de chaque offre, et du nombre maximal de négociations auxquelles il peut s'engager. En fait, l'agent doit faire un compromis entre, faire une estimation plus exacte du coût, ce qui limite le nombre de négociations à mener, et s'engager dans un nombre important de négociations, ce qui entraîne une estimation des coûts moins exacte.

d- Saturation du processus de traitement des messages

La congestion du processus de traitement des messages est un problème majeur dans la négociation des agents. Un agent peut facilement devenir saturé, c'est à dire, passer tout son temps dans la délibération des messages qui, à l'expiration du délai de réponse, deviennent temporellement désuets et ne conduisent plus à un contrat.

Plusieurs solutions ont été évoquées pour résoudre ce problème parmi lesquelles on peut citer :

- *l'adressage focalisé (focused addressing)* [Smi88a] qui signifie que dans les situations critiques les agents avec des ressources libres annoncent leur disponibilité, alors que dans les situations moins critiques se sont les agents ayant des tâches qui annoncent leurs tâches.
- *une limite de l'audience (audience restriction)* [San93] qui signifie qu'un agent ne peut envoyer des annonces qu'à un ensemble limité de contractants potentiels.
- *la supervision mutuelle (mutual monitoring)* [San96a] où un agent peut contrôler le flux de messages envoyé par les autres, et ainsi il peut pénaliser le plus impatient d'entre eux.

Le problème avec ce mode de contrôle, c'est que les agents ont besoin d'un mécanisme de motivation et de punition. Ce qui n'est pas évident avec des systèmes ouverts et des

agents à intérêts personnels. De tels agents envoient des messages tant que cela est profitable pour leurs solutions locales. Avec des réseaux ouverts, comme l'Internet, la congestion risque de devenir inévitable. Une façon d'éviter une telle situation est de limiter la charge de communication que peut faire un agent ou d'imposer des frais de traitement sur les offres. Cette dernière méthode offre l'avantage d'implanter un mécanisme dynamique d'auto-sélection qui est plus viable avec les agents autonomes et rationnels.

8.2 Extensions futures

a- Extensions possibles sur le modèle de négociation

Deux principales directions peuvent être considérées pour l'extension du modèle. D'une part, l'amélioration des mécanismes de construction des propositions en introduisant de nouveaux paramètres : (1) le paramètre temps pour l'évaluation des offres (fonction d'utilité temporelle : l'utilité associée à une offre peut diminuer ou augmenter avec le temps), (2) la notion de pénalité pour la construction des contrats (à chaque contrat les agents spécifient une valeur de pénalité en cas de résiliation, cette valeur peut elle aussi être fonction du temps pour contrôler les retards). D'autre part, l'amélioration des mécanismes de persuasion pour introduire une argumentation basée sur la démonstration de vérité des croyances (jusqu'à maintenant, on a seulement exploité les relations de dépendances et le raisonnement par cas sur l'historique des contrats).

b- Extensions possibles sur MIAMAP

Plusieurs simplifications ont été faites pendant l'implantation de MIAMAP, comme l'implantation d'un seul manager pour tous les domaines d'activités de la place commerciale. Toutes ces simplifications peuvent être corrigées dans une version future de MIAMAP. Ce travail nous a fait aussi ressortir d'autres questions intéressantes concernant le fonctionnement de la place commerciale :

- Comment fournir un mécanisme de paiement sécuritaire et confidentiel ?
- Comment contrôler les fraudes et les mauvaises représentations ?

- Comment décourager la contre-spéculation et la formation de collusions dans les enchères ?
- Comment offrir des services qui nécessitent une ouverture sur d'autres plates-formes de commerces électroniques ?

c- Futurs travaux d'expérimentation

Pendant les expérimentations faites, nous avons limité le nombre d'agents à six à cause de la capacité de la machine de travail (un PC avec 16 Meg de Ram et processeur de 166Mhz). Par conséquent, ces simulations ne donnent aucune idée sur la stabilité du système, qui réellement doit fonctionner avec plusieurs dizaines d'agents qui s'activent et se désactivent de façon dynamique. Une expérimentation sur la stabilité du système reste nécessaire pour la validation du modèle.

d- Nécessité d'un langage de négociation

Le processus de coopération d'agents intelligents est très complexe et implique plusieurs processus : processus d'identification de partenaire et de communication qui peut être supporté par un langage de communication comme KQML, processus d'influence qui nécessite un métalangage et processus de médiation et de coordination qui nécessite peut être un langage de commande. Ainsi, un langage de coopération qui permet de capter et représenter de façon explicite les connaissances aux différents niveaux de coopération est nécessaire. Les travaux qui ont été fait par M. Barbuceanu et M. S. Fox [Barbuceanu 1996a, Barbuceanu 1996b, Barbuceanu 1998a, Barbuceanu 1998b] sur le langage COOL (un métalangage construit sur KQML et KIF) constituent une grande étape dans ce sens.

e- Questions ouvertes

Les questions relatives à la manière de faire coopérer des agents autonomes et d'assurer l'équilibre entre la rationalité et la sociabilité dans des systèmes multiagent opérationnels restent ouvertes. Comme nous l'avons signalé lors de la présentation des travaux similaires (cf. 4.3), les concepts de rationalité et de sociabilité deviennent très complexes avec les systèmes multiagent, puisqu'on veut faire la coordination d'entités autonomes. Les travaux faits dans le domaine des sciences humaines constituent une source intéressante pour la construction de modèles de coopération puisqu'ils supportent

l'échange d'information imprévisible et cela dans des environnements où les agents sont hétérogènes et où chacun raisonne selon sa perspective locale de la situation.

« La chose la plus importante est de continuer à se poser des questions »

- Albert Einstein.

Références bibliographiques

[**ACTS 1997**] ACTS's projects : http://www.infowin.org/ACTS/PROJECTS

[**Adler 1989**] M. R. Adler, A. B. Davis, R. Weihmayer, R. W. Worrest. Conflict resolution strategies for nonhierarchical distributed agents. In Distributed artificial intelligence Volume II (L. Gasser & M.N. Huhns), pp:139-161, Pitman/Morgan Kaufmann Publishers,1989.

[**Agha 1988**] G. Agha and C.E. Hewitt. Concurrent programming using actors : exploiting large scale parallelism. In Readings in distributed artificial intelligence (A.H. Bond & L. Gasser), pp : 161-168, Morgan kauffmann publishers, 1988.

[**Aubuchon 1997**] N. Aubuchon. The anatomy of persuasion. New York AMACOM Publishers, 1997.

[**Barbuceanu 1996a**] M. Barbuceanu, M. S. Fox. Capturing and modeling coordination knowledge for multi-agent systems. In International Journal of Cooperative Information Systems, pp:275-314, June & September 1996.

[**Barbuceanu 1996b**] M. Barbuceanu, M. S. Fox. Coordinating multiple agents in the supply chain. In The Fifth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 96), pp:134-142, Stanford Univ., IEEE Computer Society Press, 1996.

[**Barbuceanu 1998a**] M. Barbuceanu. Conversational Agent Coordination Language. User's Guide to the Coordination Language. Disponible à : http://www.eil.utoronto.ca/ABS-page/ABS-intro.html

[**Barbuceanu 1998b**] M. Barbuceanu. Detailed documentation of the protocols. Disponible à : http://www.eil.utoronto.ca/ABS-page/ABS-intro.html

[**Benjamin 1995**] R. Benjamin, R. Wigand. Electronic markets and virtual value chains on the information superhighway, Sloan Management Review, pp: 62-72. 1995.

[**Bernard 1998**] J.C. Bernard, L. Esmahi, D. Gauvin, H. Marchal. Un Modèle de coopération pour les agents sociaux. Colloque International sur les Nouvelles

Technologies de la Répartition, NOTER'98, pp: 215-232. Montréal, Québec, Canada, Octobre 1998.

[**Bestougeff 1989**] H. Bestougeff, and G. Ligozat. Outils logiques pour le traitement du temps. Editions Masson, 1989.

[**Bigus 1997**] J. P. Bigus, J. Bigus. Constructing Intelligent Agents With Java : A Programmer's Guide to Smarter Applications . John Wiley & Sons publishers, 1997.

[**Birrel 1984**] A. Birrel and B.J. Nelson. Implementing Remote Procedure Calls. ACM Transactions on Computer Systems, pp:39-59, February 1984.

[**Bisiani 1987**] R. Bisiani, F. Alleva, A. Forin, R. Lerner, and M. Bauer. The architecture of the AGORA environment. In Distributed artificial intelligence (M.N. Huhns), pp : 99-107, Morgan kauffmann publishers, 1987.

[**Bond 1988**] A.H. Bond, and L. Gasser. An analysis of problems and research in DAI. In Readings in distributed artificial intelligence, pp : 3-35, Morgan kauffmann publishers, 1988.

[**Bowman 1994**] C. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, M. F. Schwartz. The harvest information discovery and access system. In the proceedings of the second international world wide web conference, pp :763-771, chicago, illinois, october 1994.

[**Brooks 1990**] R. A. Brooks. Elephants don't play chess. In Robotics autonomous systems, vol 6, pp : 3-15, 1990.

[**Burckert 1991**] H. Burckert, J. Muller. RATMAN : Rational Agents Testbed for Multi-Agent Networks. In Decentralized A.I.3, Proceedings of MAAMAW'91, pp: 217-230, 1991.

[**Cammarata 1983**] S. Cammarata, D. Mc-Arthur, and R. Steeb. Strategies of cooperation in distributed problem solving. In Proceedings of the 8th international joint conference on artificial intelligence, pp : 767-770, 1983.

[**Carley 1998**] K.M. Carley, M.J. Prietula, Z. Lin. Design versus cognition: the interaction of agent cognition and organizational design on organizational performance. Journal of artificial societies and social simulation, vol. 1, No 3, 1998. Disponible à http://jasss.soc.surrey.ac.uk/1/3/4.html

[**Castelfranchi 1990**] C. Castelfranchi. Social power : a point missed in multiagent, DAI and HCI. In Decentralized A.I. (MAAMAW'89), pp:49-62, 1990.

[**Castelfranchi 1992**] C. Castelfranchi, M. Miceli, A. Cesta. Dependance relations among autonomous agents. In Decentralized A.I.-3 (MAAMAW'91), pp: 215-227, 1992.

[**Castelfranchi 1995**] C. Castelfranchi. Guarantees for autonomy in cognitive agent architecture. In Intelligent agents (M. Wooldridge, N.R. Jennings), pp: 56-70, 1995.

[**Castelfranchi 1998**] C. Castelfranchi. Through the Minds of the Agents. In Journal of Artificial Societies and Social Simulation vol. 1, no. 1, 1998. Disponible à : http://www.soc.surrey.ac.uk/JASSS/1/1/5.html

[**Cavendon 1997**] L. Cavendon, A. Rao. Teamwork via team plans in intelligent autonomous agent systems. In Proceedings of worldwide computing and its applications WWC'97, pp: 106-120, March 1997.

[**Chaib-draa 1996**] B. Chaib-draa. Interaction Between Agents in Routine, Familiar and Unfamiliar Situations. In International Journal of Intelligent & Cooperative Information Systems, pp:01-25, Vol.5, N°1, 1996.

[**Chess 1995**] D. Chess et al.. Itinerant Agents for Mobile Computing. IEEE Personal Communications, Vol. 2, n° 5, pp:34-49, October 1995.

[**CLIMATE 1997**] CLIMATE, http://www.fokus.gmd.de/research/cc/ima/climate

[**Coleman 1973**] J. Coleman. The mathematics of collective action. H.E.B. Publishers, 1973.

[**Conry 1991**] S.E. Conry, K. Kuwabara, and V. R. Lesser. Multistage Negotiation for Distributed Constraint Satisfaction. In IEEE Transactions on systems, man, and cybernetics, pp : 1462-1477, N°6, Vol 21, 1991.

[**de Bono 1971**] E. de Bono. Lateral Thinking for Management, A handbook of creativity. American Management Association, 1971.

[**Debra 1994**] P. Debra, R. Post. Information retrieval in the world wide web : making client-based searching feasible. Proceedings of the first international world wide web conference. Geneve, 1994. Disponible à : http://www1.cern.ch/PapersWWW94/reinpost.ps

[**Demazeau 1989**] Y. Demazeau, and J.P. Müller. Decentralized A.I. Proceedings of the first european workshop on modelling autonomous agents in a multi-agent world, pp : 3-13, august 1989.

[**Denenbourg 1991**] J.L. Denenbourg, A. Sendova-Franks, A. Detrain, L. Chretien. The dynamics of collective sorting robot-like Ants and Ant-like robots. In From animals to animats, pp : 356-363, MIT Press, 1991.

[**Dewey 1999**] A. M. Dewey and R. Bolton. Virtual Enterprise and Emissary Computing Technology. In International Journal of Electronic Commerce, pp: 45-64, Volume 4, Number 1, Fall 1999.

[**Drexler 1988**] K. E. Drexler and M. S. Miller. Incentive engineering for computational resource management. In B. A. Huberman, editor, The Ecology of Computation, pp: 231-266. North-Holland, 1988.

[**Drogoul 1992**] A. Drogoul, J. Ferber. Multi-agent simulation as a tool for modeling societies : application to social differenciation in Ant colonies. In Artificial social system, MAAMAW'92, pp : 3-23, 1992.

[**Durfee 1987**] E.D. Durfee, V.R. Lesser, and D.E. Corkill. Cooperation throught communication in a distributed problem solving network. In Distributed artificial intelligence (M.N. Huhns), pp : 29-58, Morgan kauffmann publishers, 1987.

[**Durfee 1989**] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Trends in coopeartive distributed problem solving. In Transactions on knowledge and data engineering, 1(1), pp : 63-83, March 1989.

[**Durfee 1994**] E.H. Durfee, and J.S. Rosenshein. Distributed problem solving and multiagents systems : Comparisons and examples. In Proceedings of the 13th international workshop on distributed artificial intelligence, pp:94-104. Seattle, USA, 1994.

[**ebay**] Web site : http://listings.ebay.com/aw/listings/list/category177/index.html

[**Edmonds 1994**] E.A. Edmonds, L. Candy, R. Jones, B. Soufi. Support for collaborative design : Agents and emergence. Communications of the ACM, Vol.37, N°7, pp : 41-47, 1994.

[**Ephrati 1991**] E. Ephrati, J.S. Rosenschein. The Clarke Tax as a consensus mechanism among automated agents. In Proceedings of the Ninth National Conference on Artificial Intelligence, pp:173-184, Anaheim, California, July 1991.

[**Ephrati 1992**] E. Ephrati, J.S. Rosenschein. Constrained intelligent action: Planning under the influence of a master agent. In Proceedings of the Tenth National Conference on Artificial Intelligence, pp:263-268, San Jose, California, July 1992.

[**Ephrati 1996**] E. Ephrati, J. S. Rosenschein. Deriving Consensus in Multi-agent Systems. Journal of Artificial Intelligence, pp: 21-74, No 87, 1996.

[**Esfandiari 1996**] B. Esfandiari, G. Deflandre and J. Quinqueton. An interface agent for network supervision. In, Intelligent Agents in Telecommunications Applications - Basics, Tools, Languages and Applications (ECAI'96), pp: 21-28. edited by S. Albayrak. Publisher Amsterdam: IOS Press, 1998.

[**Esmahi 1993**] L. Esmahi. Systèmes intelligents d'aide à la décision: un cas de la gestion budgétaire. Thèse DESS, Analyste concepteur. Département d'informatique, Institut National de Statistique et d'Économie Appliquée, Maroc, 1993.

[**Esmahi 1996**] L. Esmahi, J.C. Bernard. L'interaction Humain-Ordinateur par agents. Rapport technique ISBN=EPM/RT-99/14, École polytechnique de Montréal, département de génie électrique et génie informatique, 1996.

[**Esmahi 1997**] L. Esmahi, J. C. Bernard, D. Gauvin. Étude des protocoles de coopération dans les systèmes multiagents. Rapport technique ISBN=EPM/RT-99/15, École polytechnique de Montréal, département de génie électrique et génie informatique, 1997.

[**Esmahi 1999a**] L. Esmahi, P. Dini.Une approche persuasive pour le raisonnement sociale dans les systèmes multiagents. Rapport technique No CRIM-99/07-107, ISNB=2-921316-27-7, Centre de Recherche informatique de Montréal, Mai, 1999.

[**Esmahi 1999b**] L. Esmahi, P. Dini, J.C.Bernard. Towards an Open Intelligent Market Place for Mobile Agents. The Eighth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE'99), pp : 279 -286, 16-18 June, 1999, Stanford university, California, USA.

[**Esmahi 1999d**] L. Esmahi, P. Dini, J.C. Bernard. Mediating conflicts in a Virtual Market Place for Telecommunication Network Services. The Fifth Bayona workshop on emerging technologies in telecommunications, pp: 248-254 , September 6-8, 1999, Parador Conde de Gondomar, Bayona, Spain.

[**Esmahi 1999e**] L. Esmahi, P. Dini, J.C. Bernard. A negotiation approach for mobile agents used in managing telecommunication network services. The first international workshop on Mobile Agents for Telecommunication Applications, MATA'99, pp: 455-476. 6-8 October 1999, Ottawa, Canada.

[**Esmahi 1999c**] L. Esmahi, J.C. Bernard. Socially committed agents : an approach based on CBR and dependency relations. Technical report No CRIM-99/09-132, ISNB=2-921316-54-4, Centre de Recherche informatique de Montréal, September, 1999.

[**Esmahi 2000a**] L. Esmahi, P. Dini, J.C. Bernard. MIAMAP : A virtual market place for intelligent agents. The 33rd Hawaii International Conference on System Sciences (HICSS-33), Maui, Hawaii, January 4-7, 2000.

[**Esmahi 2000b**] L. Esmahi, J. C. Bernard. Mediating conflict in mobile agents interactions : an approach based on CBR and dependencies, Submitted to the IEE Internet Computing Journal. Special on Agent Technology and the Internet, (March/April 2000).

[**Falchuck 1997**] B. Falchuck and A. Karmouch. AgentSys: A Mobile Agent System for Digital Media Access and Interaction on an Internet. In Proceedings of the IEEE Globecom'97, vol. III, pp: 1876-1880, Phoenix, USA, November 1997.

[**Ferber 1996**] J. Ferber. Reactive distributed artificial intelligence : principales and applications. In, Foundations of distributed artificial intelligence, pp : 287-314, 1996.

[**Fikes 1995**] Network-Based Information Broker. R.. Fikes, R. Engelmore, A. Farquhar, W. Pratt. Knowledge System Laboratory, Stanford University. Disponible à : http://logic.stanford.edu/cit/cnet-papers.html

[**Finin 1994**] T. Finin, J. Weber. ``Specification of the KQML Agent-Communication Language", DARPA Knowledge Sharing Initiative External Interfaces Working Group, February 9, 1994. Disponible à : http://www.cs.umb..edu/kqml/papers .

[**Finin 1997**] T. Finin, Y. Labrou, J. Mayfield. KQML as an agent communication language. In *Sofware agents,* Bradshaw J. ed., pp:291-316, MIT Press, Cambridge, 1997.

[**FIPA 1997a**] Foundation for Intelligent Physical Agents (FIPA), Agent Communication Language. FIPA'97 Draft Specification : Part2, cf. 7.3.6., June 1997. Disponible à : http://drogo.cselt.it/fipa/spec/fipa97/fipa97.htm

[**FIPA 1997b**] Foundation for Intelligent Physical Agents, http://www.cselt.it/fipa

[**FIPA 1998**] FIPA 98 specification. Disponible à : http://drogo.cselt.it/fipa/spec/fipa98/fipa98.htm

[**Fisher 1981**] R. Fisher, W. Ury. Getting to Yes: Negotiating Agreement without Giving-in. Houghton Mifflin, 1981.

[**Fisher 1994**] M. Fisher. Representing and Executing Agent-Based Systems. In Proceedings of the Intelligent Agents. ECAI-94 Workshop on Agent Theories, Architectures and Languages (ATAL'94), pp:307-323, LNAI n°890, SpringerVerlag (Ed), Amsterdam, the Netherlands, August 1994.

[**Fishburn 1970**] P. C. Fishburn. Utility theory for decision making. Wiley publisher, New York, 1970.

[**Fisher 1996**] K. Fisher, J.P. Muller, M. Pischel. Cooperative transportation scheduling : an application domain for DAI. Int. Journal of applied artificial intelligence. No 10, Vol. 1, pp: 1-33, 1996.

[**Fox 1981**] M. S. Fox. An organisational view of distributed systems. IEE Transaction on systems, man and cybernetics, pp : 70-80, vol 11, No1, january 1981.

[**Franklin 1996**] S. Franklin and A. Graesser. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, pp:21-36, Springer-Verlag, 1996.

[**Fudenberg 1991**] D. Fudenberg, J. Tirole. Game theory. MIT Press, c1991.

[**Gasser 1987**] L. Gasser, C. Braganza, R. Herman. MACE : a flexible testbed for distributed AI research. In Distributed artificial inteiligence (M.N. Huhns), pp : 119-152, Morgan kauffmann publishers 1987.

[**Gauvin 1995**] D. Gauvin. Un environnement de programmation orienté agent. troisièmes journées francophones sur l'intelligence artificielle distribuée et les systèmes

multiagents, St-Baldoph, Savoie, France, 15-17 mars 1995. Disponible à: http://www.crim.ca/sbc/english/lalo/.

[**Gauvin 1996**] D. Gauvin, H. Marchal, C. Saldanha. LALO : an open multi-agent systems development environment. In Canadian artificial intelligence (Fall 1996, special on agents), 1996.

[**Genesereth 1986**] M. R. Genesereth, M. L. Ginsberg, J. S. Rosenschein. Cooperation without communication. In Proceedings of the Fifth National Conference on Artificial Intelligence, pp 51-57, Philadelphia, PA, August 1986.

[**Genesereth 1992**] M. R. Genesereth, R. E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, March 1992.

[**Georgeff 1983**] M. Georgeff. Communication and interaction in multi-agent planning. In AAAI'83, pp : 125-129, Washington DC, 1983.

[**Gmytrasiewicz 1991a**] P. J. Gmytrasiewicz, E. H. Durfee, D. K. Wehe. "The Utility of Communication in Coordinating Intelligent Agents." In Proceedings of the Ninth National Conference on Artificial Intelligence, pp :166-172, July 1991.

[**Gmytrasiewicz 1991b**] P. J. Gmytrasiewicz, E. H. Durfee, D. K. Wehe. "A Decision-Theoretic Approach to Coordinating Multiagent Interactions." In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp : 62-68, August 1991.

[**Gruber 1992**] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In the International Workshop on Formal Ontology, Nicola Guarino, Ed., Padova, Italy, 1992. Disponible à: http://ksl-web.stanford.edu/KSL_Abstracts/KSL-93-04.html

[**Gruber 1993**] T. R. Gruber, A translation approach to portable ontology specifications. In Knowledge acquisition, vol 5, No 2, pp: 199-220, 1993.

[**Guther 1999**] A. Guther, M. Zell. MIAMI Platform Enhancement Requirements. Disponible à : http://www.fokus.gmd.de/research/cc/ima/miami/public/ac338/doc/

[**Halpern 1985**] J. Y. Halpern, Y. Moses. A guide to the modal logics of knowledge and belief: a preliminary draft. Proceedings of the IJCAI-85, pp: 480-490, Los Angeles, CA. 1985.

[**Hanselman 1996**] D. Hanselman, B. Littlefield. Mastering MATLAB : a comprehensive tutorial and reference. Prentice Hall, 1996.

[**Harrison 1995**] C. G. Harrison, D. M. Chess, A. Kershenbaum. Mobile Agents: Are they a Good Idea?. IBM T.J. Watson Research Center, March 1995, site : http://www.research.ibm.com/massdist

[**Hewitt 1993**] C. E. Hewitt. Some requirements for mobile distributed telecomputing architecture. In Artificial social systems, pp : 259-270. Springer verlag 1993.

[**Hogg 1997**] L. M. Hogg, N. R. Jennings. Socially Rational Agents. In Proc. AAAI Fall symposium on Socially Intelligent Agents, pp: 61-63. Boston, Mass., November 8-10, 1997.

[**Huberman 1995**] B. Huberman, S. H Clearwater. A multi-agent system for controlling building environments. In Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), pp :171-176, San Francisco, CA, June 1995.

[**IDC 1999a**] International Data Corporation. Worldwide internet services market and trends forcast, 1998-203. International Data Corporation's report (IDC #W18904), May, 1999.

[**IDC 1999b**] International Data Corporation. The global market forcast for internet usage and commerce. International Data Corporation's report (IDC #W19262), June, 1999.

[**IKV 1997**] IKV++. Grasshopper. Disponible à : http://www.ikv.de/products/grasshopper/index.html

[**Jennings 1992**] N.R. Jenning, T. Wittig. ARCHON : Theory and pratice. In Distributed artificial intelligence : Theory and praxis (N.M. Avouris & L. Gasser), pp : 179-195, Kluwer Academic Press, 1992.

[**Jennings 1993**] N.R. Jenning, J.A. Pople. Design and implementation of ARCHON's coordination module. In Proceedings on cooperating knowledge based systems, pp : 61-82, Keele UK, 1993.

[**Jennings 1997**] N. R. Jennings, J. R. Campos. Towards a Social Level Characterization of Socially Responsible Agents. In IEE Proceedings on Software Engineering, 144 (1), pp: 11-25, 1997.

[**Jennings 1998**] N. R. Jennings, M. J. Wooldridge. Applications of Intelligent Agents. In Agent Technology: Foundations, Applications, and Markets (eds. N. R. Jennings and M. Wooldridge), pp: 3-28, 1998.

[**Kalenka 1999**] S. Kalenka, N. R. Jennings. Socially Responsible Decision Making by Autonomous Agents. Proc. Fifth Int. Colloq. on Cognitive Science, pp: 135-149. San Sebastian, Spain, 1999.

[**Kay 1990**] A. Kay. User interface, A personal view. In The art of human computer interface design (B. Laurel). pp : 191-207, Addison Wesley, 1990.

[**Keller 1995**] A. M. Keller. Smart Catalogs and Virtual Catalogs. Computer science departement, Stanford University. Disponible à : http://logic.stanford.edu/cit/cnet-papers.html

[**Khedro 1993**] T. Khedro, M. R. Genesereth. Progressive negotiation: A strategy for resolving conflicts in cooperative distributed multidisciplinary design. In Proceedings of the Conflict Resolution Workshop, IJCAI-93, Chambery, France, September 1993.

[**Kolodner 1989**] J. Kolodner, C. Reisbeck. Case Based Reasoning. Tutorial MA2 in the eleventh international joint conference on artificial intelligence (IJCAI'89), Detroit, Michigan, USA, 1989.

[**Kolodner 1993**] J. Kolodner. Case-Based Reasoning. Morgan Kaufmann Publishers, 1993.

[**Konolige 1980**] K. Konolige, N.J. Nilson. Multiple agent planning systems. In Proceedings of the 1st national conference on artificial intelligence, pp : 138-141, august 1980.

[**Kotz 1997**] D. Kotz et al.. Agent TCL: Targeting the Needs of Mobile Computers, IEEE Internet Computing, Vol. 1, n° 4, pp:58-67, July-August 1997.

[**Kozbe 1996**] B. Kozbe et al.. The Requirements for Personal Mobile Assistants in a Mobile Telecommunication Environment. In, Intelligent Agents in Telecommunications

Applications - Basics, Tools, Languages and Applications (ECAI'96), pp: 117-123. edited by S. Albayrak. Publisher Amsterdam: IOS Press, 1998.

[**Kraus 1990**] S. Kraus, J. Wilkenfeld. The function of time in cooperative negotiations. In Proceedings of the Ninth National Conference on Artificial Intelligence, pp: 179-184, Anaheim, California, July 1990.

[**Kraus 1995**] S. Kraus, J. Wilkenfeld and G. Zlotkin Multiagent Negotiation Under Time Constraints, Artificial Intelligence journal, Vol:75, No 2, pp : 297-345, 1995.

[**Labrou 1994**] Y. Labrou, T. Finin. A semantic approach for KQML - A general purpose communication language for software agents. Proceedings of the 3rd International Conference on Information Knowledge Management, November 1994. Disponible à: http://www.csee.umbc.edu/~jklabrou/publications.html .

[**Labrou 1996**] Y. Labrou. Semantics for an agent communication language. PhD thesis dissertation submission, University of Maryland Graduate School, Baltimore, September, 1996.

[**Labrou 1997**] Y. Labrou. T. Finin. A Proposal for a new KQML Specification, TR CS-97-03, February 1997. Disponible à : http://www.cs.umbc.edu/kqml

[**LALO 1997**] Langage Agent Logiciel Objet. Tutoriel disponible à l'adresse : http://www.crim.ca/sbc/english/lalo/tutorial/t1sbra.html.

[**Lange 1998a**] D. B. Lange, M. Oshima. Programming and Deploying Java™ Mobile Agents with Aglets™. Addison-Wesley, August 1998.

[**Lange 1998b**] D. B. Lange, M. Oshima. Mobile Agents with Java: The Aglet API. World Wide Web Journal, 1998. Disponible à : http://www.genmagic.com/asa/danny

[**Lenat 1988**] D. B. Lenat. BEINGS : knowledge as interacting experts. In Readings in distributed artificial intelligence (A.H. Bond & L. Gasser), pp : 161-168, Morgan kauffmann publishers, 1988.

[**Lesser 1975**] V. R. Lesser, L. D. Erman. A Multi-level organisation for problem solving using many diverse, cooperating sources of knowlwdge. In Proceedings of the International Joint Conference on Artificial Intelligence, pp: 483-490, 1975.

[**Luce 1967**] R. D. Luce, H. Raiffa. Games and decisions, John Wiley & Sons, 1967.

[**MacKie-Mason 1994**] J. K. MacKie-Mason, H. R. Varian. Some FAQs about usage-based pricing, 1994. Disponible à: ftp://alfred.sims.berkeley.edu/pub/Papers/useFAQs.html

[**Maes 1996**] P. Maes, A. Chavez. Kasbah: An Agent Marketplace for Buying and Selling Goods. In Conference on Practical Applications of Intelligent Agents and Multi-Agent Technology. April 1996. Disponible à:
http://agents.www.media.mit.edu/groups/agents/publications/

[**Magedanz 1996**] T. Magedanz, K. Rothermel, S. Krause, Intelligent Agents : an Emerging Technology for Next Generation Telecommunications. In Proceedings of the IEEE Infocom, pp: 464-472. San Francisco, USA, March 1996.

[**Magedanz 1999**] T. Magedanz, C. Baumer, M. Breugst, S. Choy. GrassHopper-a Universal Agent Platform Based on OMG MASIF And FIPA Standards. In the First International Workshop on Mobile Agents for Telecommunication Applications (MATA'99), pp:01-18, Ottawa, Canada, 6-8 October 1999.

[**Malone 1988**] T. Malone, R. Fikes, M. Hoard. Entreprise : a market-like task scheduler for distributed computing environments. In the ecology of computation (B.A. Huberman), pp :177-205, Elsevier science publishers,1988.

[**MASIF 1998**] The MASIF standardization, OMG TC Document. Disponible à: http://www.fokus.gmd.de/research/cc/ima/masif/documents.html

[**Matwin 1989**] S. Matwin, S. Szpakowicz, Z. Koperczak, G.E. Kersten, W. Michalowski. Negoplan: an expert system shell for negotiation support. IEEE Expert, pp:50-61, Winter 1989.

[**Merz 1996**] M. Merz and W. Lamersdorf. Agents, Services and Electronic Markets: How do they Integrate?. In Proceedings of the IFIP/IEEE International Conference on Distributed Platforms (ICDP'96), Dresden, Germany, March 1996. Disponible à: http://vsys-www.informatik.uni-hamburg.de/papers/ps/icdp96/icdp-96.ps.gz

[**Minsky 1985**] M. Minsky. The society of mind. Simon and schuster, New York, 1985.

[**Muller 1996**] H. J. Müller. Negotiation principales. In Foundations of distributed artificial intelligence (G.M.P. O'Hare, N.R. Jennings), pp: 211-230, 1996.

[**Nash 1950**] J. Nash. Equilibrium points in n-person games. In proceedings of the national academy of science, vol : 36, pp :48-49, 1950.

[**Negroponte 1995**] N. Negroponte. Being Digital. Published by Alfred A. Knopf, Inc, 1995.

[**Nicol 1993**] J. R. Nicol, T. Wilkes, F. A. Monola. Object orientation in heterogenous distributed computing systems. IEE computer, pp : 57-67, june 1993.

[**Nygren 1996**] K. Nygren, I. M. Jonsson, O. Carlvik. An Agent System for Media on Demand Services. First International Conference on Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96), Londres, UK, April 1996. Disponible à:http://www.fek.su.se/forskar/program/imorg/dok/1996-08-28_1/er-medlab/mediaagent/paam_enkel.doc.html

[**Objectspace 1997**] Objectsapce. Voyager platform. Disponible à : http://www.objectspace.com/products/vgrOverview.htm

[**Oliveira 1996**] R. Oliveira, J. Labetoulle. From Intelligent Agents towards Management by Request. In Proceedings of the 4th International Conference on Intelligence in Networks (ICIN'96), pp: 30-34, Bordeaux, France, November 1996.

[**Omari 1999**] S. Omari, R. Boutaba. Policy-based Control Agents for Boundary Routers in Differentiated Services IP. In the First International Workshop on Mobile Agents for Telecommunication Applications (MATA'99), pp:477-490, Ottawa, Canada , 6-8 October 1999.

[**OMG 1994**], Object Management Group, http://www.omg.org

[**OMG 1999**] OMG Unified Modeling Language Specification, 1999. Disponible à : http://www.rational.com/uml/index.jtmpl

[**OnSale**] Web site : http://www.onsale.com/helpinfo/firsttime/about.htm?source=splash_14

[**Orfali 1998**] R. Orfali, D. Harkey. Client/server programming with Java and CORBA. Wiley Computer Publisher, New York, 1998.

[**Perret 1996**] S. Perret, A. Duda. Mobile assistant programming for efficient information access on the WWW. In Proceedings of the Fifth International World Wide Web Conference, Paris, France, 1996. Disponible à: http://www5conf.inria.fr/fich_html/papers/P42/Overview.html

[**Ponsard 1977**] J. P. Ponsard. Logique de la négociation et théorie des jeux. Editions d'organisation, 1977.

[**Pruitt 1981**] D. G. Pruitt. Negotiation behavior. Academic press, 1981.

[**Raifa 1982**] H. Raifa. The Art and Science of Negotiation. Harvard University Press, 1982.

[**Rasmusen, 1989**] E. Rasmusen. Games and Information. Basil Blackwell, 1989.

[**RCC 1999**] Retail Council of Canada. The comprehensive report on E-Retail in Canada. The Retail Council of Canada and IBM, June 1999.

[**Reicken 1994**] D. Reicken. A conversation with Marvin Minsky about agents. In Communication of the ACM, pp : 23-29, Vol 37, N° 7, 1994.

[**Reinhardt 1994**] A. Reinhardt. The Network with Smarts. Byte Magazine, October 1994. Disponible à: http://www.byte.com/art/9410/sec7/art1.htm

[**Rodriguez 1997**] J.A. Rodriguez, F. Noriega, C. Sierra, J. Padget. FM96.5 – A java based electronic auction house. In second international conference on the practical application of intelligent agents and multi-agent technology, PAAM'97, 1997.

[**Rosenschein 1985**] J. S. Rosenschein, M. R. Genesereth. Deals among rational agents. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, pp : 91-99, Los Angeles, California, August 1985.

[**Rosenschein 1994**] J. S. Rosenschein, G. Zlotkin. Rules of Encounter. MIT Press, 1994.

[**Russel 1995**] S. J. Russel, P. Norvig. Artificial intelligence : A modern approach. Prentice-Hall, 1995.

[**Sandholm 1993**] T. W. Sandholm. An implementation of the Contract Net protocol based on marginal cost calculations. In proceedings 11th national conference on artificial intelligence, pp :256-262, 1993.

[**Sandholm 1995**] T. W. Sandholm, V. R. Lasser. Issues in automated negotiation and electronic commerce : Extending the Contract Net framework. In proceedings of the first international conference on multi-agent systems (ICMAS-95), pp :328-335, San Francisco 1995.

[**Sandholm 1996a**] T. W. Sandholm. Limitations of the Vickrey auction in computational multiagent systems. In Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96), pp: 299-306, Keihanna Plaza, Kyoto, Japan, December 1996.

[**Sandholm 1996b**] T. W. Sandholm. Negociation among self-interested computationally limited agents. PhD Thesis, University of Massachusetts, 1996.

[**Sathi 1989**] A. Sathi, M. S. Fox. Constraint-Directed Negociation of resources allocations. In Distributed artificial intelligence Volume II (L. Gasser & M.N. Huhns), pp : 163-194, Pitman/Morgan Kaufmann Publishers,1989.

[**Selker 1994**] T. Selker. Coach : A teaching Agent that learns. In Communication of the ACM, pp : 92-99, Vol 37, N° 7, 1994.

[**Shoham 1993**] Y. Shoham. Agent-oriented programming. In Artificial Intelligence, pp : 51-92, N°1, vol 60,1993.

[**Singh 1994**] M. P. Singh. Multiagent system : a theoretical framework for intention know-how, and communications. Springer-Verlag, 1994.

[**Smith 1981**] R. G. Smith, R. Davis. Frameworks for cooperation in distributed problem solving. In IEEE transactions on systems, man and cybernetics, pp : 61-70, N°1, Vol 11, 1981.

[**Smith 1988a**] R. G. Smith. The Contract Net Protocol : high-level communication and control in a distributed problem solver. In Readings in distributed artificial intelligence (A.H. Bond & L. Gasser), pp : 357-366, Morgan kauffmann publishers, 1988.

[**Smith 1988b**] R. G. Smith, R. Davis. Negociation as a metaphor for distributed problem solving. In Readings in distributed artificial intelligence (A.H. Bond & L. Gasser), pp : 333-356, Morgan kauffmann publishers, 1988.

[**Smith 1999**] K. D. Smith, R. B. Paranjape. Mobile Web Agents for Telemedicine. In the First International Workshop on Mobile Agents for Telecommunication Applications (MATA'99), pp:405-418, Ottawa, Canada, 6-8 October 1999.

[**Sproull 1978**] R. F. Sproull, D. Cohen. High-level Protocols. In Proceedings of the IEEE, pp :1371-1386, N°11, Vol 66, 1978.

[**Straber 1997**] M. Straßer, J. Baumann, F. Hohl. Mole - A Java Based Mobile Agent System. In M. Muhlhausser (ed.), special issues in object oriented programming, pp:301-308. Verlag 1997.

[**Sycara 1985**] K. Sycara. Arguments of Persuasion in Labour Mediation. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85), Vol 1, pp:294-296, Los Angeles, Ca., August 1985.

[**Sycara 1987**] K. Sycara. Finding Creative Solutions in Adversarial Impasses. In the Proceedings of the Ninth Annual Conference of the Cognitive Science Society, pp: 997-1003, Seattle, Washington, July 1987.

[**Sycara 1988a**] K. Sycara. Using Case-Based Reasoning for plan adaptation and repair. In Proceedings of the 1988 Case-Based Reasonning Workshop, pp : 425-434, May, 1988.

[**Sycara 1988b**] K. Sycara. Resolving Goal Conflicts via Negotiation. In Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88), pp:245-250, St. Paul. MN., August 1988.

[**Sycara 1993**] K. Sycara. Machine Learning for Intelligent Support of Conflict Resolution. Decision Support Systems, Vol. 10, pp:121-136, 1993.

[**Tennenholtz 1989**] M. Tennenholtz, Y. Moses. On copperation in a multi-entity model. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), pp : 918-923, Detroit, Michigan USA, 1989.

[**Tennenhouse 1997**] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, G. J. Minden. A Survey of Active Network Research. IEEE Communications Magazine, Vol. 35, No. 1, pp: 80-86. January 1997.

[**Tsvetovatyy 1997**] M. Tsvetovatyy, M. Gini, B. Mobasher, Z. Wieckowski. MAGMA: An Agent-Based Virtual Market for Electronic Commerce. In Applied Artificial Intelligence, special issue on Intelligent Agents, No 6, September 1997. Disponible à: http://www-users.cs.umn.edu/~gini/papers/magma.pdf .

[**Villinger 1997**] K. Villinger, C. Burger. Generic mobile agents for electronic market, University of Stuttgart, 1997.

[**VME**] Virtual Market Enterprises . Web site : http://vme.net/

[**Waldspurger 1992**] C. A Waldspurger, T. Hogg, B. Huberman, J. O. Kephart, W. S. Stornetta. Spawn: A distributed computational economy. IEEE Transactions on Software Engineering, 18(2), pp:103-117, 1992.

[**Werkman 1990**] K. J. Werkman. Multiagent cooperative problem solving through negociation and perspective sharing. PhD Thesis, Lehigh University, 1990.

[**Werkman 1993**] K. J. Werkman. Negociation in DAI as an infrastructure component for collaborative entreprises. In proceedings of the IEEE second workshop on enabling technologies infrastructure for collaborative entreprises, pp: 104 -117. April 1993.

[**Werner 1992**] E. Werner, Y. Demazeau. Decentralized AI3, North Holland, June 1992.

[**White 1996**] J. E. White. Telescript Technology: Mobile Agents. General Magic Inc., January 1996, disponible à : http://www.genmagic.com/Telescript/Whitepapers

[**Wong 1999**] C. D. Wong, J. DiCelie. Mobile Agent Development Using Concordia. Tutorial in the First International Workshop on Mobile Agents for Telecommunication Applications (MATA'99), Ottawa, Canada, 6-8 October 1999.

[**Wooldridge 1994**] M. Wooldrige, N.R. Jennings. Towards a theory of cooperative problem solving. In Proceedings of the 6th European workshop on modelling autonomous agents in a multi-agent world MAMAW'94, pp : 15-26. Odense, Danmark, 1994.

[**Yemini 1991**] Y. Yemini. Network Management by Delegation, in Integrated Network Management II, Krishnan and Zimmer (Eds.), Elsevier Science Publisher, 1991.

[**Zeuthen 1930**] F. Zeuthen. Problems of monopoly and economic welfare, 1930.

[**Zlotkin 1989**] G. Zlotkin, J. S. Rosenschein. Negotiation and task sharing among autonomous agents in cooperative domains. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp: 912-917, Detroit, MI, August 1989.

[**Zlotkin 1990**] G. Zlotkin, J. S. Rosenschein. Negotiation and conflict resolution in non-cooperative domains. In Proceedings of the Eighth National Conference on Artificial Intelligence, pp: 100-105, Boston, Massachusetts, July 1990.

[**Zlotkin 1991**] G. Zlotkin, J. S. Rosenschein. Cooperation and conflict resolution via negotiation among autonomous agents in noncooperative domains. IEEE Transactions on Systems, Man and Cybernetics, 21(6), pp:1317-1324, November/December 1991.

[**Zlotkin 1993a**] G. Zlotkin, J. S. Rosenschein. A domain theory for task oriented negotiation. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, pp: 416-422, Chambery, France, August 1993.

[**Zlotkin 1993b**] G. Zlotkin, J. S. Rosenschein. The extent of cooperation in state-oriented domains. Computers and Artificial Intelligence, 12(2), pp:105-122, 1993.

[**Zlotkin 1993c**] G. Zlotkin, J. S. Rosenschein. Negotiation with incomplete information about worth: Strict versus tolerant mechanisms. In Proceedings of the First International Conference on Intelligent and Cooperative Information Systems, pp: 175-184, Rotterdam, The Netherlands, May 1993.

[**Znaty 1997**] S. Znaty et M. P. Gervais. Les réseaux intelligents : ingénierie des services de télécommunication, Editions Hermès, pp: 279, 1997.

[**Zwass 1999**] V. Zwass. Structure and macro-level of electronic commerce : from technological infrastructure to electronic MarketPlaces. In Foundations of Information Systems. Vladimir Zwass, Fairleigh Dickinson University, 1999. Disponible à : http://www.mhhe.com/business/mis/zwass/ecpaper.html .

Annexe I

Dans cette annexe, nous introduisons quelques concepts de la notion d'utilité que nous utilisons dans différentes parties de cette thèse. Ces concepts sont : ***la fonction d'utilité, le point de rupture et le point de Nash.***

9.1 Fonction d'utilité

En théorie des jeux, on suppose toujours que chaque agent (joueur) dispose d'une fonction d'utilité qui lui permet d'évaluer les conséquences de ses choix et des choix adverses en regard de ses propres objectifs.

Définition [Fishburn 1970] *: soit $(C_k P_k)$ k=1,..., n la conséquence incertaine susceptible d'entraîner l'une ou l'autre des conséquences certaines C_k avec une probabilité P_k $(P_k \geq 0)$, $\forall k \sum^n_{k=1} Pk=1$. L'utilité de l'agent Ai pour la conséquence $(C_k P_k)$ est :*

$f_i[(C_k P_k)\ k=1,..., n] = \sum^n_{k=1} P_k * f_i[C_k]$, où $f_i[C_k]$ est la valeur qu'associe l'agent A_i aux choix C_k.

Propriété *: la notion de **fonction d'utilité** pour un agent A_i n'est définie qu'à une transformation affine près : si f_i est une fonction d'utilité, alors $\forall\ \alpha \geq 0$ et β, on a $(\alpha f_i + \beta)$ est une autre fonction d'utilité. Ces deux fonctions traduisent d'une manière équivalente les évaluations des conséquences pour un agent Ai en regard de ses objectifs.*

À titre d'illustration, voici comment un agent A_i pourrait construire sa ***fonction d'utilité*** dans le cas simple où son objectif consiste à maximiser son gain financier. Supposons que celui-ci puisse varier entre 0 et 100$. En utilisant la propriété ci-dessus, on peut toujours fixer arbitrairement l'utilité en deux points. Elle prendra donc $f_i(0\$) = 0$ et $f_i(100\$) = 1$.

Par approximation, l'agent cherchera ensuite une valeur $X_{0.5}$ comprise entre 0 et 100$ telle qu'il soit indifférent entre rester dans la même situation avec une chance sur deux ou bien recevoir 100$ avec une chance sur deux.

Il en déduira donc : $f_i(X_{0.5}) = 1/2\, f_i(0) + 1/2\, f_i(100) = 0.5$.

Si l'opération est répétée avec différents points, on obtient ***la courbe d'utilité*** pour A_i sous la forme suivante :

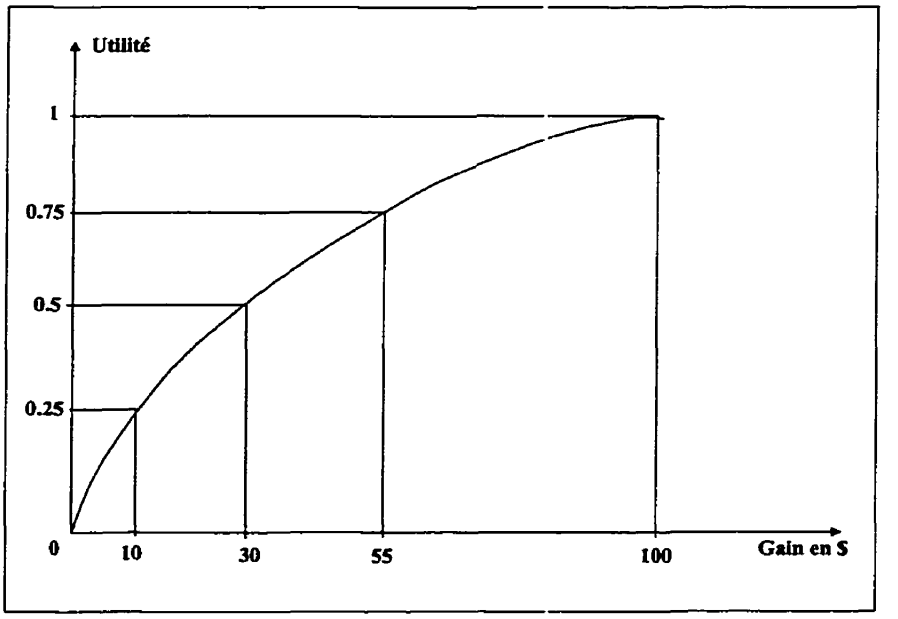


Figure 9.1 : Fonction d'utilité d'un agent

Puisqu'un agent doit évaluer non seulement sa propre utilité, mais aussi celle de ses pairs, on comprend facilement l'importance cruciale de l'information dont il dispose sur les objectifs de chacun. La décision que prendra l'agent dépend donc de l'exactitude des informations sur lesquelles il raisonne et sur la précision qu'il choisira pour son évaluation.

9.2 Point de rupture et point de Nash

Considérons un cas simple de deux agents A_1 et A_2 qui négocient la partage d'une somme d'argent P sachant que chacun des agents peut obtenir en agissant de façon indépendante de l'autre respectivement S_1 et S_2.

Cette situation peut être schématisée par la figure suivante :

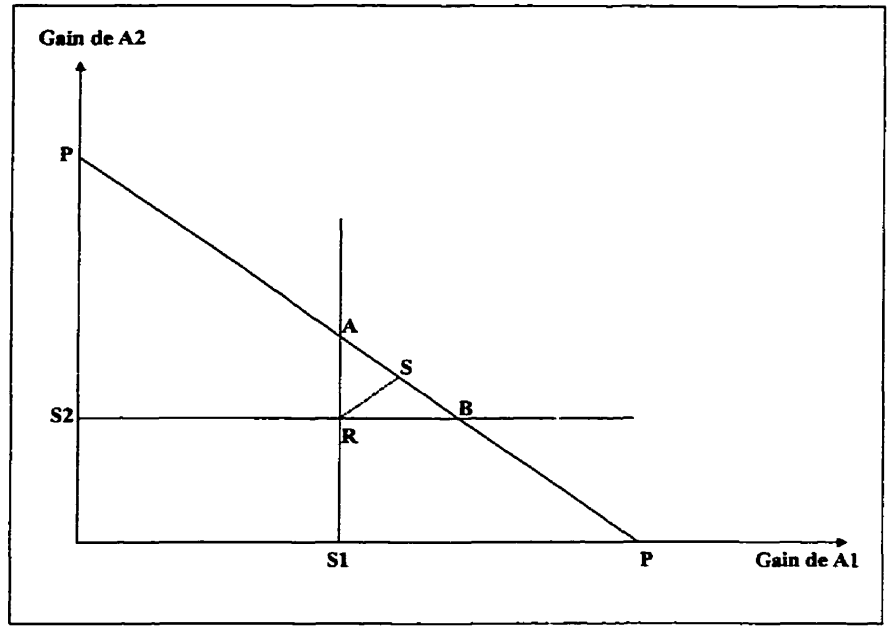


Figure 9.2 : Cas de négociation à deux agents

Sur le diagramme de la figure 9.2, le point *R* représente la situation de conflit. En fait, c'est ***le point de rupture*** ou de « statu-quo » pour la négociation [Ponssard 1977].

Les points sur le segment *PP* correspondent à toutes les répartitions possibles de la somme *P*. On comprend aisément que les seules répartitions susceptibles d'être envisagées par les deux agents sont celles du segment *AB*, puisque chacun d'eux n'acceptera un accord que s'il y gagne quelque chose. Dans ces conditions, si les deux agents sont rationnels, le résultat du partage serait le point *S* qui consiste à partager le profit de façon équitable.

Les coordonnées du point *S* sont : $[S_1 + (P - S_1 - S_2) / 2]$ pour l'agent A_1 et $[S_2 + (P - S_2 - S_1) / 2]$ pour l'agent A_2.

Cette solution ne tient pas compte du rapport de force qui peut exister entre les deux agents, et qui est relatif au poids que chaque agent accorde au point de rupture.

Par l'introduction des poids ou rapports de force, on voit bien l'intérêt stratégique pour chacun des agents d'améliorer sa position de rupture, ce qui fait augmenter sa part du profit.

Pour généraliser ce modèle, nous pouvons considérer une situation où la négociation porte sur une série d'éléments qui peuvent être évalués de façons différentes par chacun des agents. Un point mineur pour l'un peut être important pour l'autre et vice-versa.

D'une manière générale, nous supposons que chaque accord entre les agents se traduit par un certain niveau de satisfaction. Cette satisfaction s'exprime en utilité. Pour A_1, $U_1 = f_1(x_1, ...x_n)$ et pour A_2, $U_2 = f_2(x_1, ...x_n)$, où les paramètres $x_1, ...x_n$ correspondent aux différents éléments de l'accord.

Dans ces conditions, la figure 9.2 est remplacée par la figure 9.3.

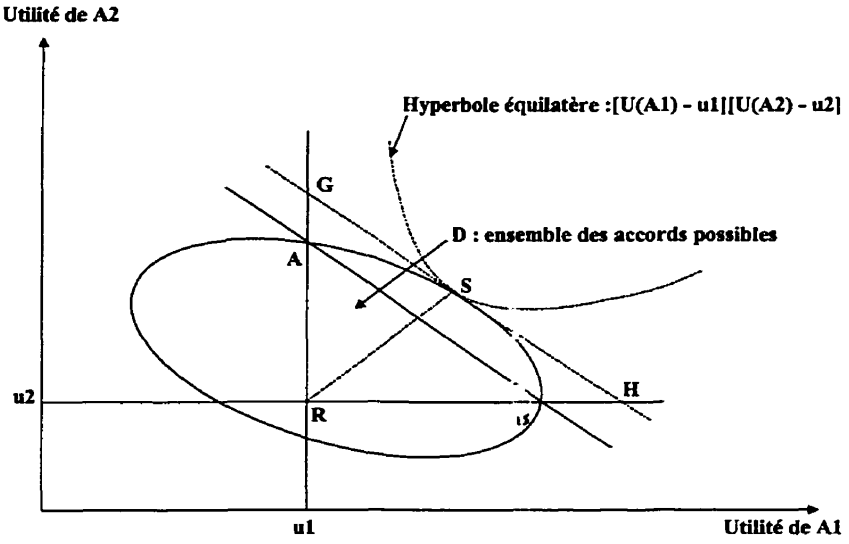


Figure 9.3 : Négociation à deux agents (cas général)

Le point de rupture R a ses cordonnées exprimées en utilité comme suit : $R = (u_1, u_2)$. Puisque chacun des agents doit gagner à la négociation, les accords envisageables se limitent en fait au triangle curviligne RAB.

Pour qu'une solution soit à la fois globalement efficace et équitable, elle doit donc en même temps maximiser $[U(A_1) + U(A_2)]$ et se situer sur la première bissectrice ayant R pour origine, c'est-à-dire sur la droite $U(A_1) - u_1 = U(A_2) - u_2$.

J. Nash [Nash 1950] avait démontré que ce point solution S n'est autre que l'intersection entre le triangle curviligne RAB et la tangente GH à D, tel que $SH = SR = SG$.

Cette solution de la négociation porte le nom de son auteur et s'appelle ***le point d'équilibre de Nash***.

Annexe II : Exemple de représentation de l'état mental de l'agent

Dans cette annexe nous essayons de rapprocher le lecteur des concepts décrits dans la figure 6.3, plus spécialement les concepts représentant l'état mental de l'agent: *Croyances, Capacités et Engagements*. Nous décrivons un exemple d'utilisation de l'état mental de l'agent pour définir le comportement d'un agent qui contrôle le bras d'un robot [LALO 1997].

10.1 Scénario

L'agent qui contrôle le bras du robot fait partie d'un ensemble d'agents : un agent wagon, le bras du robot, les pinces, etc. Dans ce scénario, les pinces sont placées à l'extrémité du bras pour réparer des câbles dans un poteau. Un ensemble de caméras permet aux différents agents d'avoir leurs cordonnées. Chaque élément de ce système est contrôlé par un agent. La réparation du câble est une tâche complexe et nécessite une coopération de la part des différents agents du système.

Dans cet exemple nous allons présenter seulement l'agent bras du robot. La position de départ du bras est *(X=0, Y=0, Z=0)*, et la décision initiale de l'agent est de se placer dans cette position. La position du bras peut être changée de deux façons : soit de façon accidentelle par un événement inattendu qui se produit dans l'environnement (i.e. présence du vent) ou de façon volontaire lorsque le bras reçoit l'ordre de changer de position.

Dans une première opération l'agent bras du robot prend la décision de faire un recalibrage toutes les 30 secondes, si nécessaire, et de recharger sa batterie tous les 5 minutes si celle-ci descend à 30% de sa charge. Si un autre agent demande au bras de faire un mouvement en lui envoyant un message KQML *(achieve(#sender, content : move(...)....))*, l'agent bras vérifie s'il peut atteindre la position demandée, sinon il envoie un message 'sorry' au demandeur et ne fait rien. S'il peut atteindre la position, l'agent

détermine un plan pour l'atteindre en calculant la matrice de coordonnées des mouvements nécessaires puis fait le positionnement demandé.

10.2 Instanciation de l'agent bras_robot

Selon le schéma conceptuel de la figure 6.3. l'agent bras_robot se définit comme suit :

Déclaration des capacités :

Tasks:

Public composite move(X, Y, Z, client); //définit la fonction d'exécuter un mouvement.

Private composite recalibrate(X, Y, Z); //fonction qui permet de repositionner le bras à la position initiale (X:0, Y:0, Z:0).

Private atomic able_to_reach(X, Y, Z, For); //fonction qui permet de déterminer si l'agent peut atteindre la position ou non.

Private atomic path_planning(X, Y, Z); //fonction qui permet de calculer la matrice des mouvements pour arriver à une position donnée.

Private atomic execute_path(X, Y, Z); //fonction qui permet la réalisation du mouvement vers position donnée.

Private composite recharge_battery(P); //fonction qui permet la réalisation du rechargement de la batterie de l'agent.

Déclaration des croyances initiales :

Beliefs:

Beliefs.add(#now : my_name(bras_agent)); //croyance de l'agent sur lui même

Beliefs.add(#now : my_x_position(X=0)); //croyance définissant la position initiale de l'agent en largeur

Beliefs.add(#now : my_y_position(Y=0)); //croyance définissant la position initiale de l'agent en longueur

Beliefs.add(#now : my_z_position(Z=0)); //croyance définissant la position initiale de l'agent en hauteur

Beliefs.add(#now : reachable(X :0, Y :0, Z :0, For : ?xx)); //croyance définissant que l'agent est capable d'atteindre la position (0, 0, 0). Cette croyance est nécessaire puisqu'elle sert comme base de calcul pour déterminer si une autre position peut être atteinte ou non.

Beliefs.add(#now : permitted_agent(name : operator1)); //croyance définissant la permission de l'agent 'operator1' à utiliser le bras.

Beliefs.add(#now : battery_level(L:0)); //croyance définissant que la batterie de l'agent est vide au départ.

Beliefs.add(#now : my_plug_power(X:-1, Y:0, Z:0)); //croyance de l'agent sur la position qu'il faut réaliser pour déclencher le chargement de sa batterie.

Declaration des engagements initiaux :

Commitments:

Commitments.add(begin_at: #now, recharge_battery(power:200); //l'agent prend au début la décision de faire recharger sa batterie pour des éventuels mouvements

Commitments.add(begin_at: #now, recalibrate(X:0, Y:0, Z:0); //l'agent prend au début la décision de faire son recalibrage à la position (0, 0, 0).

Déclaration des règles de traitement des messages :

RulesBase:

R0 : If

received(ask(sender :a1, reply_with :t1, content :?c1)

belief(#now : permitted_agent(a1)

Then

send_event(#now, reply(sender : #myname, in_reply_to :t1, content : ?c1))

Si l'agent reçoit une requête d'information et l'agent requérant possède la permission d'avoir cette information alors l'agent lui répond par un message qui contient la valeur du paramètre demandé.

R1 : If

received(achieve(sender : ag, content : beging_at t : move(X :?x, Y :?y, Z :?z)));

belief(#now :permitted_agent(ag);

Then

commitments.add(to :ag, beging_at t : move(X :?x, Y :?y, Z :?z));

Cette règle permet de générer l'engagement de l'agent bras pour examiner la possibilité de réaliser le mouvement demandé par l'agent 'ag'. Un contrôle est fait sur les permissions de l'agent à utiliser le service demandé.

Déclaration des règles d'exécution des tâches :

R2 : If

executing : move(X :?x, Y :?y, Z :?z, client :?who);

Then

decompose :

beging_at #now : recalibrate(X :?x, Y :?y, Z :?z);

beging_at #now+30 : able_to_reach(X :?x, Y :?y, Z :?z, For :?who);

Cette règle contrôle l'exécution d'engagement relatif au mouvement du bras. La réalisation du mouvement est une tâche composée qui consiste à :

- Faire d'abord un recalibrage pour repositionner le bras,
- Déterminer si le bras peut atteindre la position demandée. Si oui, l'exécution de la fonction 'able_to_reach' permet d'ajouter à la base de croyance de l'agent un fait qui entraîne le déclenchement des règles responsables de la réalisation du mouvement.

R3 : If

executing : able_to_reach(X :?x, Y :?y, Z :?z, For :?who);

Then

succeed;

Puisque la tâche 'able_to_reach' est atomique, l'instruction 'succeed' permet d'appeler la méthode qui implante le traitement de cette tâche. L'exécution de cette méthode entraîne l'ajout d'un fait à la base de croyance de l'agent qui spécifie si l'agent peut atteindre la position demandée ou non.

R4 : If

executing : recalibrate(X :?x, Y :?y, Z :?z);

Then

decompose :

beging_at #now : execute_path(X :?x, Y :?y, Z :?z);

beging_at #now+30 : recalibrate(X :?x, Y :?y, Z :?z);

Cette règle correspond à l'exécution de la tâche composée de recalibrage. La fonction 'execute_path' permet de réaliser le mouvement aux coordonnées fournies et le deuxième appel de 'recalibrate' ferme la boucle de recalibrage qui doit être faite toutes les 30 secondes.

R5 : If

executing : path_planning(X :?x, Y :?y, Z :?z);

Then

succeed;

commitments.add(to :#my_self, beging_at ?now : execute_path(X :?x, Y :?y, Z :?z));

L'instruction 'succeed' correspond à l'appel de la méthode qui implante la tâche 'path_planning'. L'exécution de cette instruction en premier force l'agent à déterminer d'abord le plan du mouvement avant de procéder à son exécution.

R6 : If

executing : execute_path(X :?x, Y :?y, Z :?z);

Committed(to :#my_self, beging_at ?now : recalibrate(X :?x, Y :?y, Z :?z));

Then

succeed;

R7 : If

executing : recharge_battery(P:?p);

belief(#now : battery_level(L:0.30));

belief(#now : my_plug_power(X:?x, Y:?y, Z:?z));

Then

decompose :

beging_at #now : recalibrate(X :?x, Y :?y, Z :?z);

beging_at #now : able_to_reach(X :?x, Y :?y, Z :?z, For :#myself);

beging_at #now+300 : recharge_battery(P:?p);

Cette règle contrôle l'exécution d'engagement relatif au rechargement de la batterie du bras. L'opération de rechargement est une tâche composée qui consiste à :

- Faire d'abord un recalibrage pour repositionner le bras,
- Exécuter la fonction 'able_to_reach' qui permet d'ajouter à la base de croyance de l'agent un fait qui entraîne le déclenchement des règles responsables de la réalisation du mouvement.
- Reprogrammer un autre rechargement, si nécessaire, après 5 minutes (300 secondes).

Déclaration des règles de raisonnement sur les croyances :

R8 : If

belief(#now :reachable(X :?x, Y :?y, Z:?z, who_asked :?ag);

Then

commitments.add(to :#my_self, beging_at ?now : path_planning(X :?x, Y :?y, Z :?z));

R9 : If

belief(#now : not reachable(X :?x, Y :?y, Z:?z, who_asked :?ag);

Then

commitments.add(to :#my_self, beging_at ?now : send_event(#now, sorry(sender : #myname, receiver :?ag, in_reply_to : achieve(sender : ag, content : beging_at t : move(X :?x, Y :?y, Z :?z))));

Une fois la tâche 'able_to_reach' est terminée, elle permet l'ajout d'une croyance 'reachable' ou 'not reachable'. La vérification de l'une de ces deux croyances permet, en conséquence, le déclenchement de l'une des règles ci-dessus. La première règle (R8) entraîne la prise d'engagement par l'agent bras à planifier le chemin du mouvement demandé. La seconde (R9) entraîne l'envoi d'un message 'sorry' à l'agent demandeur lui indiquant que le mouvement ne peut pas être exécuté.

Dans certains cas du raisonnement par règles, il faut forcer la séquence des règles pour être sur que leur déclenchement va se faire dans l'ordre convenable. Dans ce contexte la règle (R9) prend d'abord l'engagement de planifier le chemin avant la réalisation du mouvement par la règle (R5).

Définition des méthodes implantant les tâches:

Pour pouvoir réaliser les tâches atomiques, il faut définir la classe actuateur (effector cf.6.3.1) qui les implantent. Chaque tâche atomique est définie par une méthode qui définit le traitement de la tâche. À titre d'exemple, le code suivant définit les tâches 'able_to_reach' et 'execute_path'.

```
Boolean able_to_reach(string t) {
        Long longueur_max_carre;
        Position new_pos = get_position(t)
        If (new_pos.equals(null) {
                New_pos = null;
                Return false
        Else {
                Longueur_max_caree = bras.longueur[0] + bras.longueur[1];
                Longueur_max_caree = Longueur_max_caree * Longueur_max_caree;
        }
        if  (Longueur_max_caree >= (new_pos.x * new_pos.x + new_pos.y * new_pos.y + new_pos.z
        * new_pos.z)) {
                // here the position is reachable
                beliefs.add(#now :reachable(X :new_pos.x, Y : new_pos.y, Z: new_pos.z))
        }
        else {
                //here the position is not reachable
                beliefs.add(#now : not reachable(X :new_pos.x, Y : new_pos.y, Z: new_pos.z))
        }
        ...
        ...
        return True
}

Boolean execute_path_process(plan  t) {
        Position new_pos = t.get_position(t)
        If ( new_pos.x = cabine.x and new_pos.y = cabine.y and new_pos.z = cabine.z)
```

```
            // position unchanged. Do nothing.
        Else {
            Cabine.x = new_pos.x
            Cabine.y = new_pos.y
            Cabine.z = new_pos.z
        }
        ...
        ...

        Return True
}
```

Les autres concepts décrits dans la figure 6.3, comme le gestionnaire de négociation, le générateur de proposition, la table des communautés et le graphe de dépendance sont des modules spécifiques au gestionnaire de la place commerciale MIAMAP. Les rôles de ces modules se définissent comme suit:

- Gestionnaire de négociation: le manager de la place commerciale maintient une table indexée de tous les sessions de négociation et de médiation actives dans la place. Le manager contrôle aussi les messages échangés entre les agents pendant ses sessions, ce qui lui permet de détecter et d'arrêter les boucles infinis dans l'échange de messages et aussi de contrôler les ressources disponibles pour permettre à d'autres agents de démarrer d'autres sessions.
- Générateur de proposition: le manager maintient la liste des prix courants des différents services vendus dans la place. Les agents peuvent donc soit demander au manager de fournir la valeur courante pour un article donné, soit lui demander de construire une proposition complète avec tous les paramètres du contrat (prix, agent partenaire, type de contrat). Le module générateur de proposition permet donc au manager de répondre à ces requêtes et de maintenir la listes des prix courants des articles.
- Table des communautés: une communauté correspond à un domaine d'activité. Pendant leur enregistrement, les agents déclarent leur capacité de vendre un service

donné. Ceci conduit donc à leur enregistrement dans une table qui regroupe tous les agents qui marchandent sur ce service. Ces tables permettent au manager de répondre facilement aux requêtes de recommandation de partenaires des agents.

- Le graphe de dépendance et son utilité sont détaillés dans l'annexe III.

Annexe III : Exemple numérique pour la négociation en utilisant le graphe de dépendance

Dans cette annexe nous décrivons un exemple numérique de négociation des agents. Plus spécifiquement l'exemple montre l'utilisation du graphe de dépendance dans la construction de contrat par échange de but.

11.1 Scénario

Dans cet exemple nous considérons quatre agents actifs dans la place commerciale qui négocient pour la satisfaction de leurs buts qui consistent à vendre ou acheter des articles. Le tableau suivant décrit la situation qui se présente dans la place commerciale.

Tableau 11.1: scénario d'activité dans la place commerciale

Articles	Agent A1		Agent A2		Agent A3		Agent A4	
	Prix de Vente	Prix D'achat	Prix de Vente	Prix D'achat	Prix de Vente	Prix D'achat	Prix de Vente	Prix D'achat
I1	>=400					<=300		
I2		<=300			>=400			
I3					>=400			<=300
I4					>=400			<=600
I5	>=400			<=300				
I6			>=300			<=200		
I7				<=500	>=400			
I8		<=400	>=500					
I9			>=200			<=200		
I8+I9			>=600					

11.2 Construction du graphe de dépendance

Pendant leur enregistrement auprès du manager de la place commerciale, les agents déclarent leur capacité de vendre ou acheter des services. En conséquence, ceci permet au manager de construire le graphe de dépendance des agents actifs dans la place

commerciale. Pendant cette étape le manager met aussi à jour les registres de la place commerciale.

Pour déclarer sa capacité de vendre un article, l'agent envoie un message KQML avec la performative 'advertise'. Par exemple, le message suivant permet à l'agent A1 d'annoncer sa capacité de vendre des articles de type I1:

Advertise(sender: <A1>, receiver: <Manager>, content:(item:<I1>, Qos: <L2>, price:<400>, ...), language: <KQML>, ontology: <Telecom>)

Pour déclarer son désir d'achat d'un article, l'agent envoi un message KQML avec la performative 'subscribe'. Le message suivant permet à l'agent A1 d'annoncer son désir d'acheter des articles de type I2:

Subscribe(sender: <A1>, receiver:<Manager>, content:(item:<I2>, Qos: <L1>, price:<300>, ...), language: <KQML>, ontology: <Telecom>)

Le graphe est donc une structure dynamique qui varie en fonctions des agents qui sont actifs dans la place commerciale. Dans le scénario ci dessus, après l'enregistrement de tous les agents, le manager aura construit le graphe suivant:

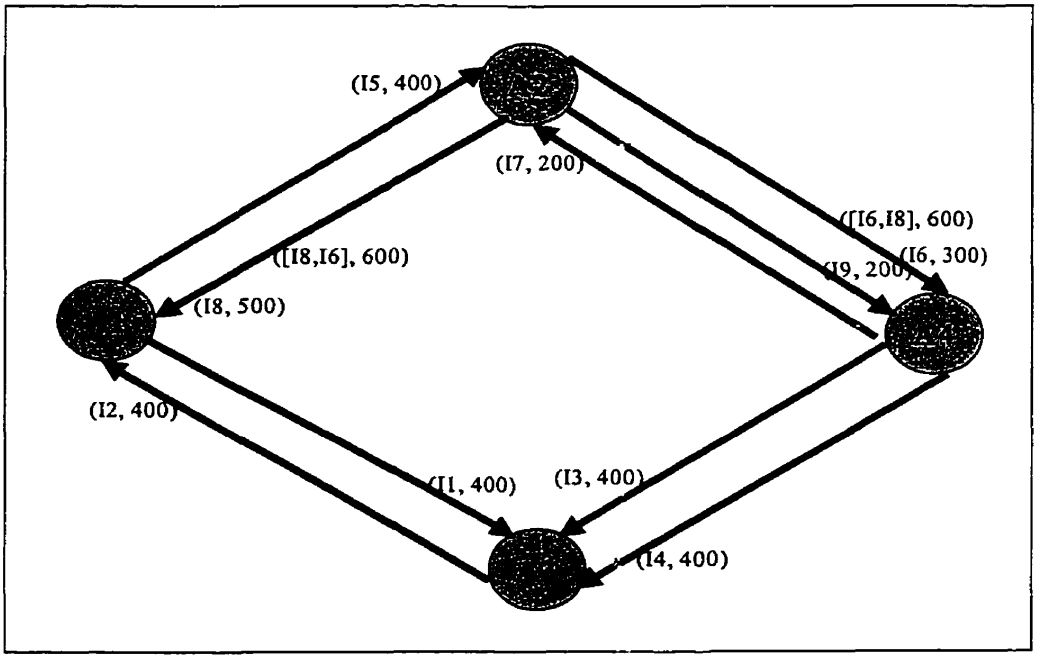


Figure 11.1: Graphe de dépendances correspondant au scénario du tableau 11.1

11.3 Exécution d'une session de négociation

Une session de négociation prend place lorsqu'un des agents sélectionne un article de sa liste d'achat puis demande au manager de lui recommander un acheteur potentiel.

Etape1:

Si on suppose par exemple que c'est l'agent A4 qui veut acheter un article de type I3. La requête adressée au manager sera de la forme:

Recommend-One(sender: <A4>, receiver:<Manager>, content:(item:<I3>, ...), in_reply_to:<nul>, reply_with: <R1>, language: <KQML>, ontology: <Telecom>).

Etape2:

Une fois la requête traitée par le manager, celui-ci répond par un message 'tell' contenant le nom du fournisseur potentiel avec lequel l'agent peut entrer en négociation. Dans cet exemple puisque le manager ne trouve qu'un seul fournisseur pour cet article (l'agent A3), alors il répond à A4 par le message suivant:

Tell(sender: <Manager>, receiver:<A4>, content:(item:<I3>, agent:<A3>), in_reply_to:<R1>, reply_with: <nul>, language: <KQML>, ontology: <Telecom>)

Etape3:

Après la réception de cette information, l'agent A4 initialise une session de négociation avec A3 en lui demandant de lui faire une proposition concernant l'article I3:

CFP(sender: <A4>, receiver:<A3>, content:(item:<I3>, Qos:<L2>, ...), reply_with:<NS1>, language: <FIPA>, ontology: <Telecom>)

Etape4:

L'agent A3 utilise alors sa fonction d'utilité pour faire une proposition à A4:

Propose(sender:<A3>, receiver:<A4>, content:(item:<I3>, Qos:<L2>, price:<480>, ...), in_reply_to: <NS1>, reply_with:<NS1>, language: <FIPA>, ontology: <Telecom>)

Etape5:

Puisque l'agent A4 ne peut accepter de payer que 300$ alors il décide de faire une contre proposition. Puisque c'était la première proposition de A3, alors il utilise seulement ses fonctions d'utilité (sans évaluation de risque) pour générer l'offre et répond par le message suivant:

Counter-Propose(sender:<A4>, receiver:<A3>, content:(item:<I3>, Qos:<L2>, price:<280>, ...), in_reply_to: <NS1>, reply_with:<NS1>, language: <FIPA>, ontology: <Telecom>)

Etape6:

L'agent A3 ne peut pas donc accepter la nouvelle l'offre de A4. En conséquence il décide de faire une contre proposition. En supposant que A3 possède la capacité d'utiliser la stratégie de négociation comme décrite dans la section 5.4, alors ce dernier demande au manager de lui donner la valeur de l'article I3 dans le marché en lui envoyant un message 'ask'. L'agent A3 utilise cette information pour estimer le risque de son partenaire.

Ask(sender:<A3>, receiver:<Manager>, content:(item:<I2>,Qos<L2>, price:<?p>), reply_with:<R2>, language: <KQML>, ontology: <Telecom>)

Etape7:

Le manager détermine le prix courant d'un article sur le marché en fonction de la valeur moyenne des transactions effectuées sur cet article. Si on suppose par exemple qu'il y ait déjà eu des transactions concernant cet article et que la valeur moyenne de ces transactions soit de 390$. Alors, le manager répond à la requête de A3 en lui envoyant le message suivant:

Reply(sender:<Manager>, receiver:<A3>, content:(item:<I2>, Qos<L2>, price:<390>), in_reply_to: <R2>, reply_with:<nul>, language: <KQML>, ontology: <Telecom>)

Etape8:

Une fois l'agent A3 reçoit l'information il procède alors à l'évaluation de son risque ainsi que celui de son partenaire comme suit:

L'utilité de A3 pour l'offre qu'il avait faite est:

$$U3(\delta(A3))=\max((480-400), 0) = 80$$

L'utilité de A3 pour l'offre de A4 est:

$$U3(\delta(A4))=\max((280-400), 0) = 0$$

L'utilité estimée de A4 pour l'offre que A3 avait faite est:

$$U4(\delta(A3))=\max((390-480), 0) = 0$$

L'utilité estimée de A4 pour l'offre qu'il avait faite est:

$$U4(\delta(A4))=\max((390-280), 0) = 110$$

Le risque de A3 est:

$$R3 = [U3(\delta(A3)) - U3(\delta(A4))]/ U3(\delta(A3)) = [80-0]/80 = 1$$

Le risque de A4 est:

$R4 = [U4(\delta(A4)) - U4(\delta(A3))]/ U4(\delta(A4)) = [110-0]/110 = 1$

Étant donné que les risques sont équivalents et que pour garantir la stabilité de la stratégie de négociation, nous avons établi la règle que seul l'agent acheteur peut répéter la même offre (cf. 5.4.1). Alors dans ce cas l'agent A3 qui est le vendeur doit nécessairement faire une concession.

Si on suppose que les agents choisissent comme degré de raffinage 1/10, alors la valeur de l'offre de A3 est déterminée par la formule suivante:

*nouvelle_offre_de_A3 = max(δ(A3)+((U3(δ(A4))*10)/(9-(R4*10))), 0);*

Si (nouvelle_offre_de_A3 <= δ(A3)) alors

{ nouvelle_offre_de_A3 = offre_maximale_de_A3}

Ce qui nous donne dans ce cas une nouvelle offre de A3 d'une valeur de 400$.

L'agent A3 répond donc à l'agent A4 par le message suivant:

Counter-Propose(sender:<A3>, receiver:<A4>, content:(item:<I3>, Qos:<L2>, price:<400>, ...), in_reply_to: <NS1>, reply_with:<NS1>, language: <FIPA>, ontology: <Telecom>)

Etape9:

L'agent A4 ne peut pas accepter cette proposition qui est supérieure à la valeur maximale qu'il peut payer, donc il décide lui aussi de répondre par une contre proposition. Si on suppose que A4 possède aussi la capacité d'utiliser la stratégie de négociation, alors celui ci procède à l'évaluation des risques comme suit:

L'utilité de A4 pour l'offre qu'il avait faite est:

$U4(\delta(A4))=\max((300-280), 0)=20$

L'utilité de A4 pour l'offre de A3 est:

$U4(\delta(A3))=\max((300-400), 0)=0$

L'utilité estimée de A3 pour l'offre que A4 avait faite est:

$U3(\delta(A4))=\max((280-390),0) = 0$

L'utilité estimée de A3 pour l'offre qu'il avait faite est:

$U3(\delta(A3))=\max((390-300),0) = 90$

Le risque de A4 est:

$R4 = [U4(\delta(A4)) - U4(\delta(A3))]/ U4(\delta(A4)) = [20-0]/20 = 1$

Le risque de A3 est:

$$R3 = [U3(\delta(A3)) - U3(\delta(A4))]/\ U3(\delta(A3)) = [90-0]/90 = 1$$

Étant donné que les risques sont équivalents et que pour garantir la stabilité de la stratégie de négociation, nous avons établi la règle que seul l'agent acheteur peut répéter la même offre. Alors dans ce cas l'agent A4 répète la même offre que précédemment sans faire de concession en envoyant le message suivant:

Counter-Propose(sender:<A4>, receiver:<A3>, content:(item:<I3>, Qos:<L2>, price:<280>, ...), in_reply_to: <NS1>, reply_with:<NS1>, language: <FIPA>, ontology: <Telecom>)

Etape10:

En recevant ce message, l'agent A3 détecte que l'agent A4 n'a plus de concession à faire et l'offre qu'il répète n'est pas satisfaisante. Alors, il décide de demander l'aide du médiateur qui est l'agent manager. Nous supposons, bien sur, que dans ce cas l'agent A4 est du type social (cf. 7.). Pour cela il envoie au manger le message suivant:

Mediate(sender:<A4>, receiver:Manager, content:(item:<I3>, ...), reply_with:<R3>, language: <FIPA>, ontology: <Telecom>)

Etape11:

Le manager consulte son registre mais ne trouve aucun autre fournisseur pour l'article I2, puis consulte le graphe de dépendance pour enfin déterminer qu'il existe une relation de multi-dépendance entre A4 et A3 sur les articles I3 et I4. En conséquence il propose à l'agents A4 une solution basée sur la réalisation d'un contrat par regroupement de buts avec A3 en combinant les articles I3 et I4 en lui envoyant le message:

Tell(sender:<Manager>, receiver:<A4>, content:(Relation: "MultiDep", agentName:"A3', itemName: "I4"), in_reply_to: <R3>, reply_with:<nul>, language: <KQML>, ontology: <Telecom>)

Etape12:

L'agent A4 reçoit alors l'information sur la relation de multi-dépendance avec A3, ce qui lui permet d'envoyer une proposition de contrat avec regroupement de I3 et I4 à A3.

Multi-Propose(sender:<A4>, receiver:<A3>, content:(item:<I3+I4>, price:<900>, ...), in_reply_to: <NS1>, reply_with:<NS1>, language: <FIPA>, ontology: <Telecom>)

Etape13:

L'agent A3 reçoit la nouvelle offre, et l'évalue par rapport à ses fonctions d'utilité. La valeur de l'offre est acceptable puisqu'elle propose 900$ qui est supérieure au minimum

acceptable pour A3, 800$ pour les deux articles. Donc A3 répond par un message 'accept'.

Accept-Proposal(sender:<A3>, receiver:<A4>, content:(item:<I3+I4>, ...), in_reply_to:<NS1>, reply_with:<NS1>, language: <FIPA>, ontology: <Telecom>)

Etape14:

L'agent A4 ferme alors la session de négociation, retire les articles de sa liste d'achat, puis envoi à A3 l'avis de contrat.

Award-Contract(sender:<A4>, receiver:<A3>, content:(item:<I3+I4>, price:<900> ...), in_reply_to: <NS1>, reply_with:<nul>, language: <FIPA>, ontology: <Telecom>)

Etape15:

L'agent A3 à son tour ferme la session de négociation et retire les articles de son inventaire.

Une autre session de négociation commence lorsque l'un des agents sélectionne un article de sa liste d'achat puis demande au manager de lui recommander un acheteur potentiel.

Dans cet exemple on peut noter aussi que l'entente sur I1 et I2 ne peut se faire que dans le cadre d'un contrat par échange de buts puisqu'il existe une relation de dépendance mutuelle entre A1 et A3 sur ces articles. De même, une entente sur I5 ou I6 ne peut se réaliser que dans le cadre d'un contrat multiagent puisqu'il existe un cycle de dépendance entre A1, A2 et A4 sur les articles I5, I6, I7, I8 et I9. Cependant, si la première session concernait soit l'article I7 ou I9, il n'y aurait pas besoin de contrat multiagent. La transaction s'effectuerait directement par un contrat simple.